PFC

Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa

Editor de Imágenes basado en Regiones. Aplicación en entorno Matlab.

Autor: Samira Hervella Azouzi
Tutor: Verónica Vilaplana
I.T.Telecomunicaciones,esp. Sonido e Imagen
EUETIT.-Terrassa

Año académico 2005-2006

...Por tu paciencia y comprensión en todos esos momentos difíciles.

Índice

Índice de Imágenes

Capítulo II.-
Representación de imágenes basadas en regiones.-

Capítulo III.-
Funcionamiento del Editor de Imágenes.-

CAPÍTULO I
OBJETIVOS Y MOTIVACIÓN

1.1.-*Objetivos.-*

El *Plan de Estudios* de la Titulación de Ingenieria Técnica de Telecomunicaciones, *especialidad Sonido e Imagen*, en la *Escuela Universitaria de Ingeniería Técnica Industial de Terrassa (EUETIT)*, incluye la asignatura obligatoria de *Procesado Digital de Imagen* en el segundo bloque de la carrera, asignatura creada por profesores del Grupo de Procesado de Señal del Departamento Teoría de la Señal y Comunicación impartida por la profesora Verónica Vilaplana.

Con el objetivo de poner en práctica los contenidos de la asignatura, se pensó en la posibilidad de generar un conjunto de herramientas y demostraciones que reforzarán las explicaciones teóricas impartidas en las clases, herramientas que podrían utilizar como material de laboratorio que permitiera a los alumnos profundizar en la materia, o como complemento de otros proyectos anteriores. Este propósito ha hecho posible la oferta del proyecto final de carrera que tiene como objetivo cubrir alguna estas necesidades.

El resultado es un *Editor de Imágenes* como herramienta para el Procesado de Imagen, en concreto para al tema de Segmentación, en el entorno Matlab.La idea inicial del proyecto, proponía la implementación de este Editor en un entorno Java, como complemento de un segundo proyecto final de carrera, generado por otro alumno de la misma titulación. Esta primera iniciativa se llevó acabo durante los primeros 6 meses que abarcaron el proceso de investigación y desarrollo del proyecto, pero factores no favorables a su desarrollo hicieron descartar esta posibilidad de generar la interface en un entorno de programación Java y reflexionar sobre cual sería el entorno más adecuado para su implementación.

La opción escogida fue buscar un entorno visual que, por un lado, fuera cómodo y fácil de utilizar,tanto para el alumno como para el profesor, y a su vez robusto y potente, con el objetivo de poder desarrollar cualquier aplicación de procesado de imagen sin limitación alguna. Lo que llevó a pensar en el entorno Matlab, conocido por sus aplicaciones en laboratorios de asignaturas de procesado de señal e imagen. La característica fundamental de este entorno es la facilidad que aporta su estructura matricial de datos. Todas las operaciones las realiza a partir de matrices (tipo base mxArray), lo que convierte a Matlab en una herramienta idónea para trabajar con imágenes. A su vez, Matlab aporta un amplio conjunto de herramientas para la creación de interficies gráficas. La creación de código resulta sencilla, ya que no es necesario preocuparse por reserva de memoria o compatibilidad de formatos entre cada una de las variables.Estas fueran los principales motivos que nos llevaron a optar por este segundo lenguaje de programación.

Para comenzar a desarrollar las herramientas, se dispuso de la versión 6.5 de Matlab. La limitación más importante de la programación en Matlab es el tiempo de ejecución de algunas de las rutinas, fundamentalmente las que utilizan bucles, muy frecuentes en procesado de imágen para recorrer imágenes. Para solventar este problema, se propone usar lenguajes de más bajo nivel que a su vez sean compatibles con Matlab. Uno de los lenguaje propuestos es C, que aunque no se ha llevado acabo en este software ,por falta de tiempo, si se ha dejado como propuesta para la creación de ejecutables de algunos algoritmos de segmentación, tal como se analiza en el apéndice de este documento.

En el capítuloII se presenta de forma breve una introducción básica de conocimientos de procesado de imagen,profundizando en la Segmentación, tema que se tratará extensamente a lo largo del desarrollo del proyecto, para que el lector pueda familiarizarse con sus estructuras y conceptos fundamentales.

1.2.-Motivación.-

La importancia del Procesado de imágen en el mundo real.-

En el Siglo 21, tenemos la implacable ola de la "Revolución de la Información". La denominada "Autopista de la Información" ha llegado ya a nuestras casas y está conectada a nuestros televisores y PCs de alto-rendimiento. El banco en casa, compra por internet, CDs de audio, DVDs... se entregan ya en nuestras casas y nos permiten el placer de ver los programas que queramos con confort. Los teléfonos de imágenes visuales nos permiten ya un tipo de comunicación que hasta ahora había sido sólo posible con un cara a cara con nuestro interlocutor. También trabajar en casa en vez de desplazarse y trabajar en el lugar de trabajo se convertirá seguramente en estos años venideros en algo muy usual.

La información consta de incontables elementos, incluidas letras, figuras, sonidos, voces, imágenes visuales e imágenes audio-visuales. En la "Revolución de la Información", toda esta información se guardará y transferirá a través de una tecnología que combina productos multimedia y técnicas de digitalización. De hecho, la competitividad internacional dependerá inevitablemente de un desarrollo que combine las fuerzas del hardware (red de comunicación super-rápida) y del software (capacidad de la red de información ultra-eficiente).

En esta era multimedia, el procesamiento de imágenes digitales es, sin lugar a dudas, la tecnología más importante que nos puede asistir en la producción de imágenes.

Los seres humanos confían fundamentalmente en el sentido de la vista. Recogemos 99 % de nuestra información a través de nuestros ojos. El procesamiento de imágenes es un proceso que concierne tanto a ordenadores como a algoritmos diversos para un propósito específico, una vez que hemos adquirido imágenes de interés a través de una cámara y / o el escáner.

Son muchas las aplicaciones que hoy en día hacen uso de las técnicas de procesado de imagen. Esta breve descripción de sus usos, nos ayudará a valorar la importancia de la mismas para la evolución de la ciencia y beneficio de la humanidad.

1.-Biología

En los campos de la biología y biomedicina, el procesamiento de imágenes se usa para analizar visualmente las muestras biológicas. Hay varios casos donde el análisis de muestras ha sido completamente automatizado usando procesamiento de imágenes. De cara a la realización de un buen análisis, se mejoran usando algoritmos como "contraste de balance" o "edge-sharpening", aquellas características inidentificables y/o poco nítidas de la imagen . La identificación automática, clasificación, categorización y análisis de ADN pueden ser realizadas por el procesamiento de imágenes.

2.-Procesado de Documentos

Una recolección y procesado automático de documentos e imágenes se presenta útil para bancos y compañías de seguros. Estos documentos están digitalmente comprimidos y guardados. Se detecta e identifica automáticamente la información impresa sobre cheques y otros documentos contables.

3.-Automatización en la industria

El procesamiento de imágenes se usa para la inspección y supervisión automática en líneas de producción de grandes empresas. Este sistema reduce mucho el error humano al tiempo que proporciona estabilidad y precisión en la producción.

4.-Diagnostico a partir de Imágenes Médicas

Los rayos X y las proyecciones CT médicas se digitalizan para examinar áreas internas del cuerpo. Un número determinado de proyecciones CT se combinan y digitalizan automáticamente para producir imágenes 3-dimensionales.

5.-Teledetección

Un satélite fotografía la corteza exterior de la Tierra a intervalos regulares y las imágenes se usan para analizar condiciones de crecimiento del cultivo, distribución de la vegetación y para exploraciones de recursos naturales.También la corteza de la Tierra puede ser convertida en un modelo 3-dimensional usando imágenes 2d tomadas por satélites.

6.- Efectos de video/film

La industria cinematográfica usa una extensa variedad de técnicas de procesado de imágenes para producir efectos visuales especiales. Las imágenes irreales y otras escenas costosas son artificialmente producidas usando técnicas de Informática Gráfica o/y Procesamiento de Imágenes Digitales.

Las principales técnicas son:

- Morphing

Morphing es una técnica de efectos especiales que visualmente transforma un material en otro de manera espontánea produciendose una transición natural. Esta técnica se usa ampliamente en películas y publicidad.

- Composiciones de imágenes

Esta técnica combina un determinado número de imágenes para conformar una sola. Muchos tipos de imágenes, tales como imágenes animadas por ordenador, pueden ser mezcladas conjuntamente.

7.-Identificación mediante biometría ,Marcas de Agua,Meteorología,Ciencia de los Materiales, ...etc.

Organización de la Memoria.-

Este documento se ha dividido en dos partes fundamentales:

- En la primera parte se expone una síntesis del estado del arte de la Segmentación, sus técnicas y algoritmos más empleados como herramientas de procesamiento de imagen, los algoritmos de Split & Merge, Crecimiento de Regiones, Watershed y K-means.
- La segunda parte se dedica al funcionamiento del software, de sus menús y herramientas generadas para el uso del mismo.

Finalmente,junto con las conclusiones se presenta un resumen de las principales aportaciones originales,y se indican interesantes propuestas futuras de investigación para la ampliación del proyecto. En el apéndice se muestra otros resultados de interés.

CAPÍTULO II
REPRESENTACIÓN DE IMAGENES BASADA EN REGIONES

REPRESENTACIÓN DE IMÁGENES BASADA EN REGIONES

2.1.- INTRODUCCIÓN.-

Los modelos de datos y señales para imágenes y secuencias de vídeo están experimentando un importante desarrollo. Parte de esta evolución se ha producido debido al elevado número de servicios multimedia a los que se debe dar soporte.Tradicionalmente, las imagenes digitales se han representado como matrices rectangulares de píxeles y los vídeos como una secuencia de frames en movimiento.Los nuevos servicios y aplicaciones multimedia requieren una representación más cercana al mundo real o que al menos tenga en cuenta parte del proceso que creó la información digital.

Mencionaremos dos ejemplos:

1. Las aplicaciones soportadas por el estándar **MPEG-4**,constituyen un caso típico: la representación basada en una matriz de píxeles no es apropiada para poder interactuar con objetos en la imagen,para codificar diferentes áreas de interés o asignar diferentes comportamientos a las entidades representadas en la imagen. En estas aplicaciones, la noción del objeto es esencial. Como consecuencia, el modelo de datos debe ser modificado y, por ejemplo, debe incluir regiones de forma arbitraria para representar al objeto. Por consiguiente, haremos una distinción entre un *objeto*, el cual visualmente tiene dos dimensiones (2-D), que es la representación de una entidad que tiene significado semántico y una *región*, la cual es una componente conexa del espacio, que generalmente viene definida por un criterio de homogeneidad. Un objeto puede estar representado por la unión de varias regiones que pueden o no estar conectadas entre si.

2. El estándar **MPEG-7** también está haciendo frente al mismo tipo de desafío.Por ejemplo, la representación de vídeo basada en secuencias de frames es inadecuada para el largo número de aplicaciones de vídeo indexado.La cantidad de complicadas funcionalidades en una aplicación de recuperación,nos hace seguir investigando en este campo.La búsqueda debe ir más allá del "fast forward" y "fast reverse" permitido por el VCRs.Uno quisiera tener acceso a una tabla de contenido del vídeo y poder saltar de una escena a otra. Esta clase de funcionalidad implica al menos una estructura de vídeo en términos de tomas individuales y escenas. Por supuesto, que la indexación y la recuperación implican también una estructuración de datos en términos de objetos, regiones, nociones semánticas,...etc.

En ambos ejemplos el nuevo modelo de datos tiene en cuenta parte del proceso de creación: una imagen es creada por la proyección de una escena visual integrada por objetos de tres dimensiones (3-D), a un plano de 2-D. Modelar la imagen en términos de regiones es conocer la proyección del objeto dentro de los límites del plano de 2-D. La detección de tomas en un vídeo tiene como objetivo encontrar lo que se ha hecho durante la edicción del video.Observese que en ambos casos, un objetivo importante del modelado de los datos es definir las transiciones espaciales y las discontinuidades temporales en una señal que tradicionalmente era vista en su totalidad. Nos referiremos a este problema como **segmentación** a lo largo de esta memoria.

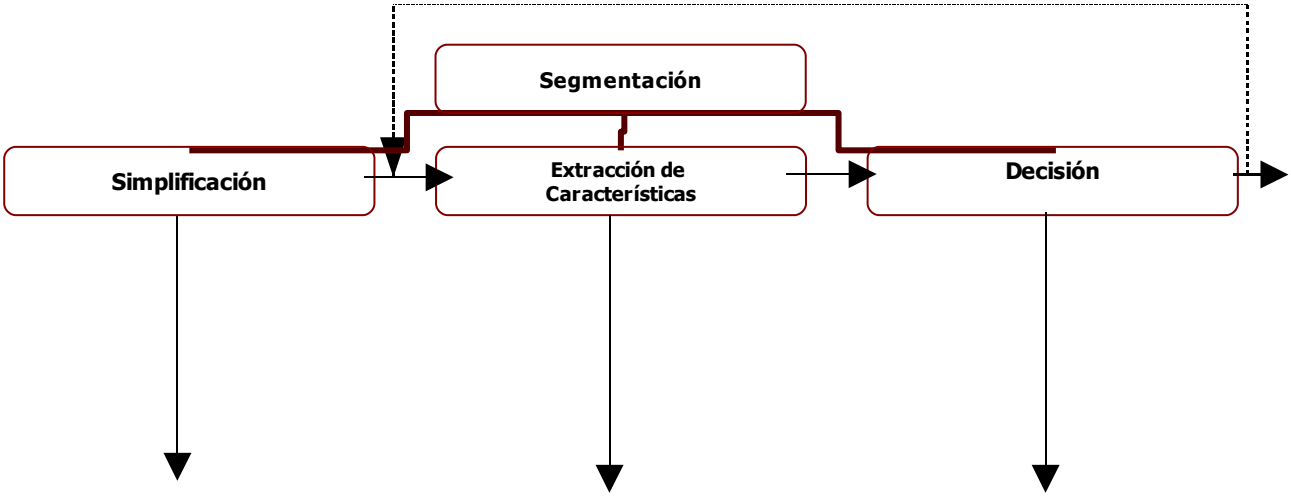
La segmentación puede ser una tarea extremadamente fácil si uno tiene acceso al proceso de producción que ha generado las discontinuidades.Por ejemplo, la generación de una imagen sintética o de un vídeo sintético implica el modelaje del mundo de 3-D y su evolución temporal. Durante la creación del mismo, es muy fácil recuperar y almacenar los contornos de los objetos de 2-D.Otro ejemplo es la edición de video, la cual crea una gran cantidad de discontinuidades espaciales o temporales.

Las *Discontinuidades espaciales* se crean combinando los objetos del primer plano, que se han filmado sobre una pantalla azul, con un fondo de secuencias, que se a tomado de forma independiente.Las *Transiciones temporales* están producidas por tomas concatenadas. En ambos casos, la detección de las discontinuidades es trivial si uno tiene acceso a la información en ese nivel de producción.

Sin embargo,si la segmentación intenta estimar que es lo que ha ocurrido durante el proceso de producción,su tarea es extremadamente difícil y hay que reconocer que el estado del arte todavía debe mejorar para lograr una segmentación robusta de imágenes y secuencias genéricas.

2.2.- ESTRATEGIAS DE SEGMENTACIÓN.-

La palabra 'segmentación' tiene un significado que depende del uso y del contexto en el cual se utilice. El objetivo básico de los algoritmos de segmentación es definir una partición del espacio.En el contexto de la imagen y del vídeo,el espacio puede ser temporal (1D), espacial (2D) o espacio-temporal (3D). En consecuencia, este espacio se llama *espacio de la decisión*. Esta sección revisa los pasos principales implicados en un algoritmo de segmentación y las principales opciones a realizar.Por otra parte, se introduce la terminología. Un esquema general para la segmentación se puede considerar como la concatenación de tres pasos principales, representados en la fig.1.



-Elimina información que no nos interesa:
-Ruido,mediante filtros Paso Bajo o de Mediana.

-Elementos pequeños, a partir de apertura o cierre. Mediante filtros morfológicos

Preserva la información de contornos.

Espacio de trabajo:
-nivel de gris
-color
-actividad
-frecuencia
-textura
-distorsión
-histograma
-DFD
-Semántica
- Movimiento
-profundidad
- ...etc

Definición de la partición:
-Clasificación
-detección de Transiciones
-detección de Regiones (Homogeneidad)

Fig.2.1.- Proceso Global de la Segmentación de Imágenes.

La simplificación.La mayoría de las veces,los datos originales en una imagen o en una secuencia de vídeo contienen informción que es irrelevante para una aplicación dada.En tales casos, los datos se deben simplificar, removiendo toda aquella información irrelevante.La simplificación controla la cantidad y naturaleza de la información que es preservada.Además los datos simplificados deben contener áreas fáciles de segmentar.Por ejemplo, la simplificación puede reducir la complejidad de texturas, ***eliminando ruido*** o ***eliminar objetos muy pequeños*** para un tamaño dado.

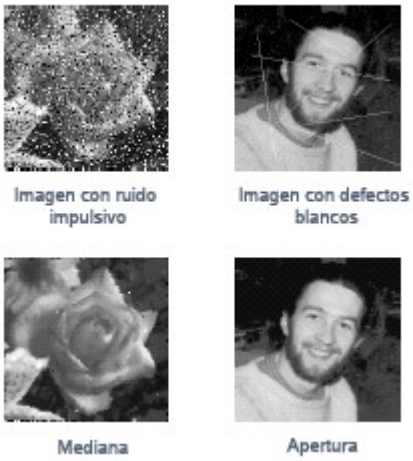


Fig.2.2.-Simplificación: Eliminación de ruido.

Las herramientas de filtrado típicas se enumeran abajo del bloque de simplificación de la fig.1.La simplificación no debe modificar la información de los contornos que es relevante para la aplicación.

Extracción de características.La segmentación se realiza sobre características específicas de los datos. La selección del espacio de características conduce al tipo de homogeneidad que se espera en la partición final. En algunas aplicaciones,los datos originales proveen directamente el espacio de características neccesario para la segmentación. Por ejemplo, para la segmentación a color, el valor del pixel puede corresponderse directamente a la característica de interés.Sin embargo, en otros casos, las carácterísticas de interés deben ser estimadas a partir de los datos originales.

Las carácterísticas típicas se enumeran debajo del bloque de la extracción de características de la Fig.1. El listado incluye actividad, frecuencia, textura, distorsión, histograma, DFD, espacios que caracterizan algunas nociones semánticas, movimiento o profundidad. En algunos casos,la estimación de las características tiene que ser hecha en una región de soporte homogénea en términos del mismo criterio que la característica dada.Consecuentemente, quizá se pueda introducir un ciclo en el proceso de segmentación de manera que la estimación dependa de los resultados de la segmentación.

*Decisión.*Finalmente para obtener una partición de los datos, el espacio de características debe ser analizado.El paso de Decisión decide sobre la posición de los contornos que forman la partición en el espacio de decisión. Los contornos separan áreas de datos que contienen elementos con las mismas características en el espacio de características selecionado. Por ejemplo en la segmentación espacial,la decisión puede producir la forma exacta de una región ,o en la segmentación temporal el grupo exacto de frames que forman la secuencia.

En la práctica, tres estratégias se pueden utilizar para definir la partición: la **Clasificación**, la D**etección de Transiciones** y la D**etección de regiones *(Homogeneidad)***.

2.3.-SEGMENTACIÓN Y CLASIFICACIÓ.-

2.3.1.- INTRODUCCIÓ

Estrategia:

- Estudio del espacio de características
- Definición de clases en el espacio de características
- Definición de las Regiones
- Casos:
  - Monodimensional:
    - 2 Clases
    - N Clases
  - Multidimensional

La Clasificación. Las técnicas de clasificación empiezan creando una partición del espacio de características y entonces traducen esta partición a una partición del espacio de decisión.
Estas técnicas han sido rara vez utilizadas para la segmentación de imagenes genéricas o vídeos porque la topología de las regiones en el espacio de decisión no se tiene en cuenta.

Recordar a modo de terminología que una región creada por un algoritmo de segmentación se define como un conjunto de elementos (pixeles o imagenes) homogéneos en el espacio de características y conexo en el espacio de decisión. Una región puede no tener ningún significado semántico. Por lo contrario, un objeto es una representación visual en 2D de una entidad con significado semántico que puede estar formado a su vez por diversas regiones.

2.3.2.-PROCESO DE CLUSTERING

Dado un conjunto de objetos, el proceso de clustering debe encontrar grupos cuyos elementos sean similares entre sí, y a la vez diferentes a los elementos de los otros grupos. Los grupos con esas características son conocidos como clusters. Los objetos son representados por *D* atributos descriptores en forma de vectores en el espacio $\mathbb{R}^D$, y con una medida de comparación de la símilitud, como la distancia, se conformarían los clusters con objetos similares. En el proceso de la conformación de los grupos, que en adelante será conocido como clustering, no existe conocimiento previo acerca de cómo se debe conformar un cluster; por tal motivo, el proceso de clustering es también conocido como *Clasificación no supervisada.*

Los tipos de objetos varían de acuerdo con el contexto de la aplicación del clustering; por ejemplo,en las tareas de clasificación dentro del *data mining*, los objetos serán registros de la base de datos; en la recuperación de la información los objetos serían documentos; y en procesamiento de imágenes los objetos serán los píxeles que conforman la imagen.

El clustering tiene múltiples aplicaciones dentro de las ciencias de la computación, como compresión de imágenes y voz digitalizadas ; en la recuperación de informaciones relacionadas ; en el *data mining*, donde se buscan grupos con ciertas características de interés (por ejemplo, descubrimiento de nuevos segmentos de clientes con el fin de mejorar los servicios que brinda una determinada empresa); en la segmentación de imágenes para dividir la imagen en regiones homogéneas (según alguna característica de interés como la intensidad, color o textura) en aplicacionemédicas ,clasificación de imágenes satelitales en zonas (urbana, descampados,bosques, ríos) .

Los métodos de clustering existentes difieren uno del otro en la forma de estructurar los clusters. Aquellos que encuentran clusters que corresponden a una partición del conjunto de

objetos se les conoce como métodos de Hard-clustering o clustering particional, siendo el más conocido el algoritmo K-Means .
A los métodos que asignan a cada objeto un valor de pertenencia con respecto a cada cluster se les conoce como métodos de Soft-clustering , y entre los más representativos de este tipo de clustering se encuentran los algoritmos Fuzzy C-Means y EM (Maximization of expectation).

2.3.3.-PROBLEMA DE CLUSTERING

Dado un conjunto de n objetos denotado por $X = \{x_1, x_2 \ldots x_n\}$, en que $x_i \in X$, sea K un número entero positivo conocido a priori.El problema del clustering consiste en encontrar una partición:

$P = \{C_1, C_2, \ldots, C_K\}$ de X, siendo C_j un cluster conformado por objetos similares, satisfaciendo una función objetivo f : $\mathbb{R}^D \rightarrow R$, y las condiciones:
$C_i \cap C_j = \varnothing$ para $i \neq j$, y $\cup C_i = X$

Para medir la similitud entre dos objetos x_a y x_b se usará una función de distancia denotada por $d(x_a, x_b)$, siendo la distancia Euclídea la más usada para medir la similitud. Así la distancia entre dos diferentes elementos $x_i = (x_i1, \ldots, x_iD)$ y $x_j = (x_j1, \ldots, x_jD)$ es:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{D} (X_{il} - X_{jl})^2}$$

Los objetos de un cluster son similares cuando las distancias entre ellos es mínima; esto permite formular la función objetivo f, como:

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} d(x_i, \bar{x}_j)^2; \qquad (1)$$

esto es, se desea minimizar (1); donde $\bar{x}_j$,conocido como elemento representativo del cluster, es la media de los elementos del cluster Cj ,

$$\underline{x}_j = \frac{1}{|C_j|} \sum x_i \in C_j,$$

y corresponde al centro del cluster. Bajo esas características,el clustering es un problema de optimización combinatoria, y ha sido demostrado que es un problema difícil..

El *algoritmo K-Means* es una técnica clustering ampliamente usada y capaz de encontrar rápidamente una solución. Dentro de los principales inconvenientes del algoritmo K-Means está su alta dependencia de la elección de los centros iniciales y su convergencia a óptimos locales. Esta última deficiencia es en realidad una debilidad de los algoritmos golosos, que son rápidos encontrando soluciones pero quedan atrapados en óptimos locales.

2.3.4.-ALGORITMO K-MEANS.CASO k- CLASES.

El algoritmo K-Means es una de las heurísticas comúnmente utilizadas para resolver el problema de clustering. La idea básica del algoritmo es obtener los k centros iniciales y formar clusters asociando todos los objetos de X a los centros más cercanos, después se recalculan los centros. Si esos centros no difieren de los centros anteriores, entonces el algoritmo termina; caso contrario, se repite el proceso de asociación con los nuevos centros hasta que no haya variación en los centros, o se cumpla algún otro criterio de parada como poco número de reasignaciones de los objetos.

A continuación se describen un algoritmo de K-means para el caso particular de trabajar con un espacio de características de dimensión igual a 1.*(Por ejemplo los niveles de gris de una imagen monocromática.)*

Suponiendo que tenemos k classes,debemos elegir k-1 umbrales y k medias de las clases,

1.- en primer lugar, elegiremos los K valores, para las medidad de cada una de las clases, (que pueden ser, por ejemplo una muestra de cada clase, esto es, un nivel de gris o un color que suponemos corresponde a cada clase).

2.-Calculamos los umbrales, como valores promedio de las muestras.

$$T_1 = (n_1+n_2)/2$$

$$\dots$$

$$T_{k-1}=(n_{k-2} + n_{k-1})/2$$

3.-Calculamos la media de cada una de las clases, a partir de los umbrales anteriores

$$n_1 = \sum_{0\le n\le T_1} nh(n) / \sum_{0\le n\le T_1} h(n)$$

$$\dots$$

$$n_1 = \sum_{T_{k-1}\le n\le M} nh(n) / \sum_{T_{k-1}\le n\le M} h(n)$$

4.-Repetimos los pasos 2 y 3, hasta que se cumpla algún criterio de terminación, como por ejemplo que la diferencia entre los dos valores de T_i consecutivos sea menor que cierto valor, para todos los umbrales T_i.

Los principales inconvenientes del algoritmo K-Means son: su sensibilidad a la inicialización (debido a esto, el resultado final depende del estado inicial); el K-Means es un algoritmo de búsqueda local (el algoritmo minimiza la función objetivo dada, pero no garantiza una configuración óptima de los clusters); y se debe tener conocimiento previo del valor que presentan los investigadores, quienes discuten cuatro métodos para la inicialización del algoritmo K-Means:

-el primer método de inicialización es completamente aleatorio;
-el segundo es el método de Forgy ;
-el tercero es el método de McQueen;
-y finalmente, el método de Kauman y Rousseeuw.

Los tres primeros métodos son, de alguna manera, aleatorios; sólo el algoritmo propuesto por Kauman y Rouseeuw es un algoritmo heurístico que identifica los objetos más representativos que prometen tener en su alrededor gran cantidad de objetos. Algunos autores concluyen diciendo que la inicialización completamente aleatoria y la propuesta de Kauman y Rouseeuw ofrecen una mejor inicialización para el algoritmo K-Means que el resto de métodos, haciéndolo más robusto.

2.4.- SEGMENTACIÓN POR DETECCIÓN DE TRANSICIONES

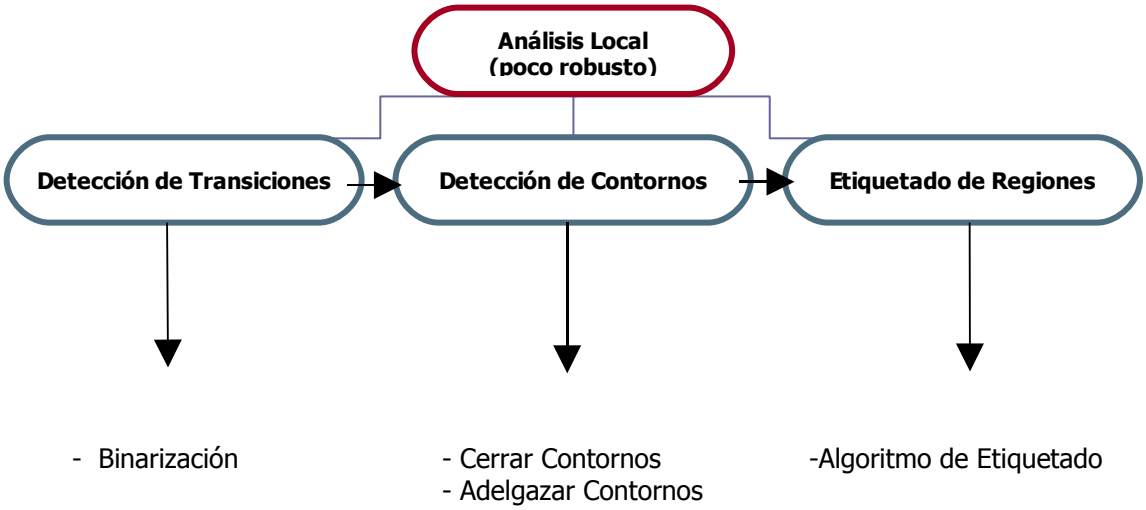


Fig.2.3.- Diagrama de Bloques de la Detección de Contornos

2.4.1.-INTRODUCCIÓ.-

Descripción de los Algoritmos de decisión. En un proceso de segmentación,el paso de la decisión debe proporcionar la partición final de los datos originales. Las técnicas basadas en Transición proponen estimar la posición de las discontinuidades en el espacio de decisión. Estas discontinuidades, son evaluadas en el espacio de características. Estas discontinuidades son realzadas por un proceso previo que se considera como un filtrado.La salida del filtro debe contener valores altos en aquellas posiciones de los datos que están cerca de las transiciones y valores bajos en las áreas homogéneas.Para esto se utilizan filtros lineales paso alto y filtros no lineales.

Sin embargo, la estimación de las posiciones de discontinuidad no crea directamente la partición.Los elementos en las áreas de transición tienen valores correspondientes a la probabilidad de ser la posición real de la transición.Esto es debido al hecho de que las transiciones abruptas en las escena puedan ser representadas por transiciones suaves locales en la imagen o en el vídeo,lo que implicaría que los contornos no sean fáciles de localizar.Por ejemplo,las transiciones estimadas de la región en la segmentación espacial pueden tener un grosor de varios píxeles.Esta información debe ser binarizada para reducir la incertidumbre en las posiciones de transición , y si fuera necesario se deben emplear técnicas de adelgazamiento para definir los contornos estimados. Esto es, transiciones binarias son adelgazadas para obtener contornos segmentados de anchura un pixel en la segmentación espacial y de un frame en la segmentación temporal. Las técnicas de adelgazamiento deben conservar la conectividad de las transiciones binarias. Varias de estas técnicas de umbral y de adelgazamiento aparecen el los libros de Procesado de Imagen.

Finalmente, los contornos resultantes quizá no puedan satisfacer la condición de conectividad .Este es un problema común tanto en la segmentación espacial como en la espacio-temporal, donde la estimación de transiciones puede ser disconexa y representar parcialmente los

contornos espaciales de las regiones. Este problema es solucionado con el uso de operaciones morfológicas como es el cierre en la que se intentan conectar esos contornos parciales para dar lugar a la partición final.

El proceso de cierre utiliza reglas geométricas que conectan los bordes.Tales reglas pueden no ser genéricas o pueden que no conduzcan a contornos reales de los datos originales del vídeo.

La desventaja principal de los métodos basados en transiciones es la falta de robustez. En particular, tanto para la segmentación espacial como espacio-temporal,si un elemento del contorno de la región no es detectado, la región entera es fusionada con otra región. En otras palabras un error local puede llegar a tener consecuencias muy significativas.Otros puntos débiles son el Cierre de contornos y el algoritmo de adelgazamiento.En la mayoría del tiempo sólo utilizan la información binaria geométrica obtenida después del umbral. Consecuentemente,la localización de la transición después del algoritmo de adelgazamiento o del Cierre de contornos puede no ser exacta.A pesar de las desventajas previamente comentadas,algunos de estos métodos de transición espacial se han propuesto en muchos libros de procesamiento.

2.4.2.DECISIÓN DE TRANSICIONES.-

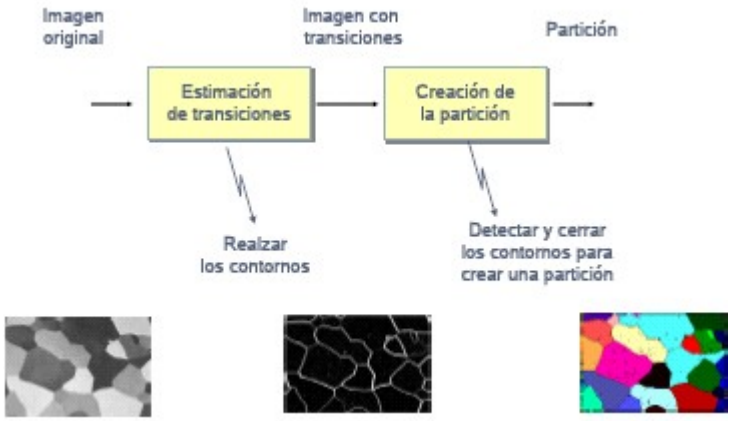


Fig.2.4.-Esquema decisión transiciones

El proceso de detección de transiciones consiste en dos pasos: un primer paso de ***estimación de transiciones***, donde partiendo de la imagen original obtenemos una imagen con transiciones. El objetivo aquí es entonces utilizar algún proceso que realce los contornos.
En segundo lugar, un paso de ***creación de la partición***, donde partiendo de la imagen con las transiciones realzadas obtenemos la partición.En este paso,el objetivo es detectar y cerrar los contornos para crear la partición.

Estimación de transiciones.-

En primer lugar habrá que decidir con qué modelo de imagen vamos a trabajar.Con el modelo de píxeles no, porque este solo tiene en cuenta el valor de cada píxel y no su entorno, y lo que

necesitamos estimar son las variaciones abruptas de valor.Podemos trabajar entonces con el *modelo espacio-frecuencia* o con el *modelo geométrico.*

En el *modelo espacio-frecuencia,* recordemos que las transiciones corresponden a las altas frecuencias. En el *modelo geométrico,* las transiciones corresponden a los saltos verticales *(mirando la imagen en tres dimensiones, valor de gris en función de la posición espacial)*.

Modelo espacio-frecuencia.-

Existen diferentes métodos para encontrar contornos con ambos modelos de imagen.Para el *modelo espacio-frecuencia*, hay operadores que aproximan *al gradiente o al laplaciano*.

Los diferentes operadores de gradiente se diferencian en el método usado para estimar los gradientes lineales (horizontal y vertical o en otras direcciones) y en como combinar esta información.Estos métodos funcionan correctamente cuando los contornos son abruptos.Si se debe detectar transiciones suaves, el nivel de gris del resultado suele ser bajo, y al aplicar el umbral los contornos se eliminan.Para detectar transiciones más suaves se puede utilizar el laplaciano (los cruces por cero forman un conjunto de curvas, en principio deberían ser cerradas; la información sobre la amplitud de la transición se guarda en el nivel de gris del resultado). Los operadores laplacianos son muy sensibles al ruido (más que los de gradiente) por lo que se debería filtrar inicialmente. Se suele filtrar previamente la imagen con un filtro gaussiano.

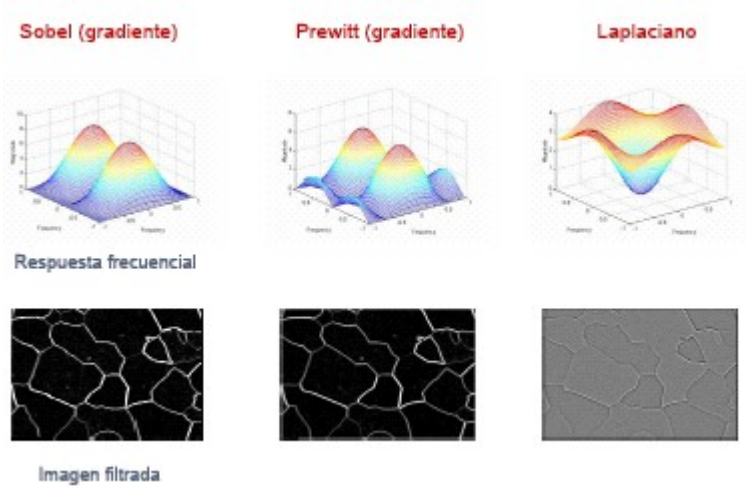


Fig.2.5.- Modelos de gradiente

Modelo geométrico.-

Para este modelo existen tres tipos de gradiente:por dilatación, por erosión y gradiente morfológico.En la figura vemos la imagen original y cada uno de estos gradientes. Recordar que el gradiente por dilatación encuentra contornos exteriores a los objetos, el gradiente por erosión encuentra contornos interiores, y el gradiente morfológico encuentra gradientes simétricos, pero dobles (de dos píxeles de ancho).

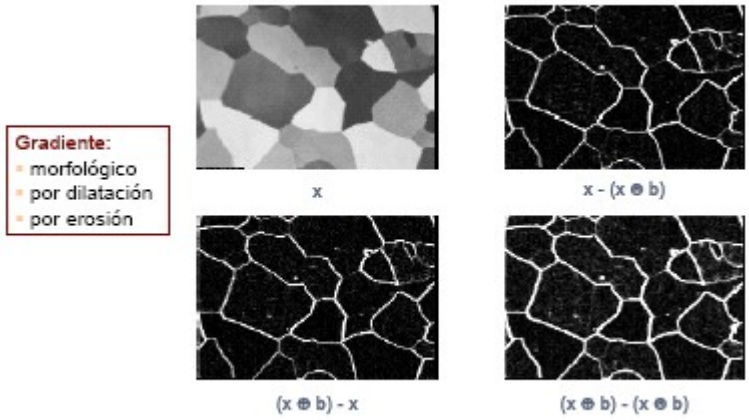


Fig.2.6.-Gradiente:morfológico, por dilatación y por erosión.

Creación de la partición.-

Una vez estimadas o realzadas las transiciones, habrá que crear la partición.Para esto tenemos que detectar las transiciones, obteniendo una imagen binarizada, y luego refinar esta detección eliminando puntos que no son de contornos, adelgazándolos o cerrando contornos abiertos, con lo cual obtenemos una imagen de contornos (detectar contornos).Finalmente habrá que crear la partición.

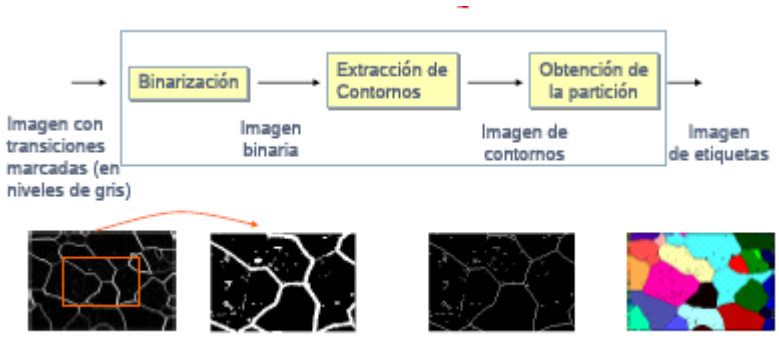


Fig2.7.-Esquema Detección de contornos.

Cuando aplicamos el umbral o detectamos cruces por cero, puede ocurrir que se detecten puntos aislados como puntos de contornos, que los contornos no estén cerrados, o que sean demasiado gruesos, y habrá que aplicar algunas técnicas para cerrar contornos o adelgazarlos.

2.4.3.-LA BINARIZACIÓN.-

Intentaremos definir de forma descriptiva cómo llegar a simplificar una imagen a nivel de gris a partir del análisis de su histograma. En muchos casos, dicha simplificación no consiste en una simple binarización, sino que se divide el histograma en diversas partes obteniéndose una imagen compuesta por más de dos niveles de gris.

El primer paso a realizar es obtener el Histograma de la imagen original, obtener un gráfico que indique cuál es la distribución, respecto del nivel de gris, que tienen todos los píxeles de la imagen. Para ello, en el gráfico se representan, por un lado, en el eje de las abcisas, todos los niveles de gris de los que consta la imagen, y por otro lado, en el eje de las ordenadas, el número de pixeles que tienen dicho nivel de gris.

A continuación se analiza dicho histograma, se toma el criterio oportuno según la simplificación que se quiera realizar, y por último se colocan unos umbrales en el histograma lo que provoca que se formen las regiones (dos en el caso de la binarización pero pueden ser más).Cada una de estas regiones queda etiquetada con un nivel de gris diferente(para la binarización. Una región con negro y la otra con blanco).En el siguiente esquema de bloques queda resumido el proceso:

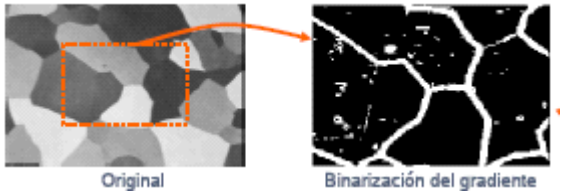


Fig.2.8.- Bloques del proceso de Binarización

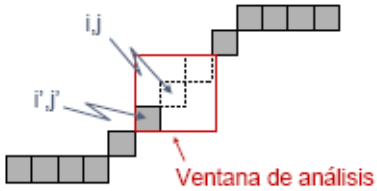
2.4.4.-CIERRE DE CONTORNOS.-

Podemos hablar de dos técnicas básicas: '*técnicas binarias*' y '*técnicas utilizando el gradiente*'.

1.-Técnicas binarias.

Una vez que tenemos la imagen binarizada, con 1's (o 255) para los puntos de contornos y 0's para el resto, una estrategia muy sencilla para cerrar contornos es aplicar un cierre a la imagen.

2.-Técnicas utilizando el gradiente.



Una de las técnicas más simples consiste en analizar las características de los píxeles en un pequeño entorno alrededor de cada punto de la imagen que ha sido etiquetado como punto de contorno. Todos los puntos que son similares de acuerdo con un conjunto de criterios predefinidos son unidos, formando un contorno de píxeles que comparten este criterio.

Las dos principales propiedades usadas para establecer estas similitudes de puntos de contornos son:

(1).-la magnitud de la respuesta del operador de gradiente utilizado para producir el pixel del contorno.
(2).-la dirección del vector gradiente.

Entonces para cada punto de coordenadas (i,j) se considera una ventana centrada en el punto.Si en la ventana ya hay un punto de contorno (i',j') tal que ambos tienen magnitudes y direcciones similares de gradiente, entonces (i,j) es etiquetado también como punto de contorno (y la dirección del borde es perpendicular a la dirección del gradiente).

Este proceso se repite para cada pixel de la imagen.Hay que llevar registro de los nuevos puntos de contorno a medida que se va moviendo el centro de la ventana de píxel a píxel (se les puede asignar un nivel de gris -diferente al de los puntos de contornos iniciales- a cada uno de ellos).

2.4.5.-ALGORITMO DE THINNING.-

Mostraremos una breve descipción de los pasos a seguir en el algoritmo.Formalmente, el algoritmo de Thinning podemos definirlo como:

$$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$$

$$\{B\} = (B^1, B^2, B^3, \ldots, B^n)$$

donde$\otimes$representa la aplicación del algoritmo de Thinning sobre la imagen A, a partir de un conjunto de elementos estructurantes B,donde B^i es una rotación de B^{i-1}.

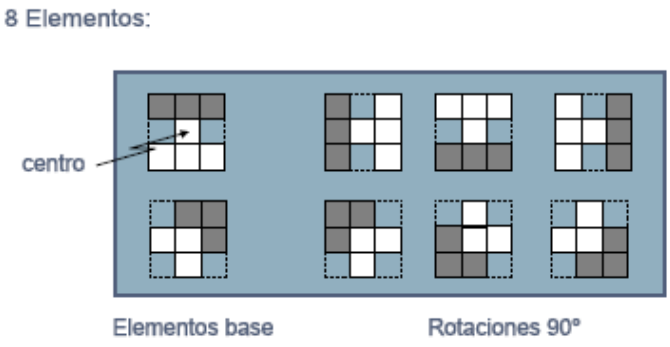


Fig.2.9.-Ejemplo de Máscara de Thinning

El algoritmo de Thinning, desde un punto de vista descriptivo,consiste en realizar sucesivos "barridos" por todos los píxeles de la imagen con los elementos del conjunto B,y comparar aquellos píxeles que pertenezcan a un contorno (1's binarios) con sus píxeles vecinos, proceso que determinará la eliminación o no de dicho píxel, y después se rota la "Matriz de Thinning".
Es importante no alterar el orden del proceso para obtener los objetivos que se pretenden.Si en lugar de analizar primero todos los píxeles, es decir, hacer un 'barrido" de toda la imagen,y a continuación rotar la "matriz de thinning" (cambiar B^i por B^{i+1}), se rota la matriz después de aplicarla sobre cada píxel, al mismo tiempo que los contornos se van adelgazando se producen roturas de los mismos.

Entonces, la idea del adelgazamiento de contornos es ir eliminando los píxeles del contorno que sobran, pero sin romperlos.

Obsevaciones:

1).-El thinning depende de la forma del conjunto y de la forma del elemento estructurante.

2).-La elección del elemento inicial y el orden de la secuencia de thinnings influyen en el resultado final.

3).-Estas transformaciones no son muy robustas. Esta falta de robustez se puede ver en la aparicion de un gran número de ramas que dependen fuertemente de la elección del elemento estructurante.

A continucación se muestra un ejemplo de los resultados obtenido en cada paso del algoritmo.Podemos observar como se han adelgazado los contornos sin producirse rotura alguna.

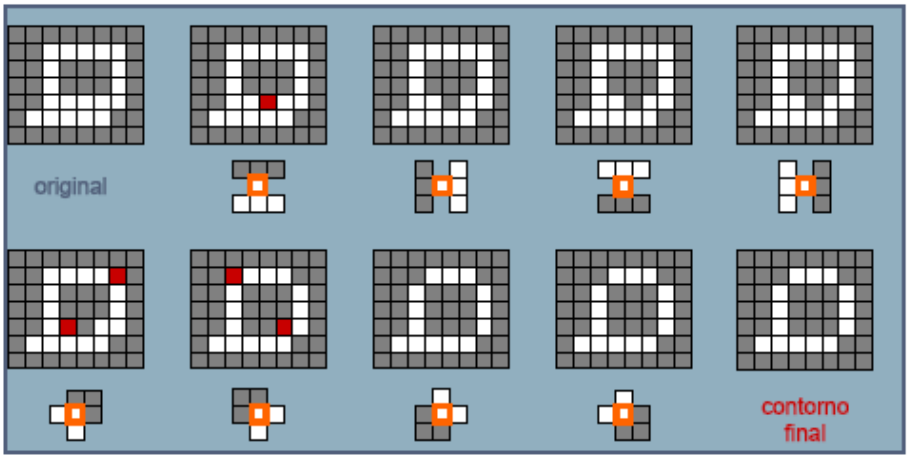


Fig.2.10.- Ejemplo Algoritmo de Thinning

2.5.- SEGMENTACIÓN BASADA EN HOMOGENEIDAD

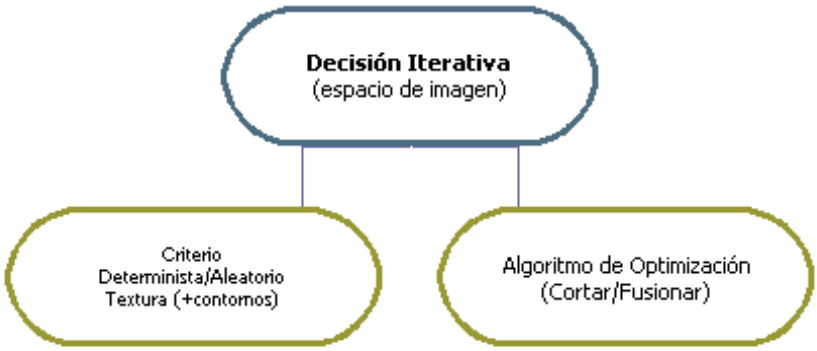
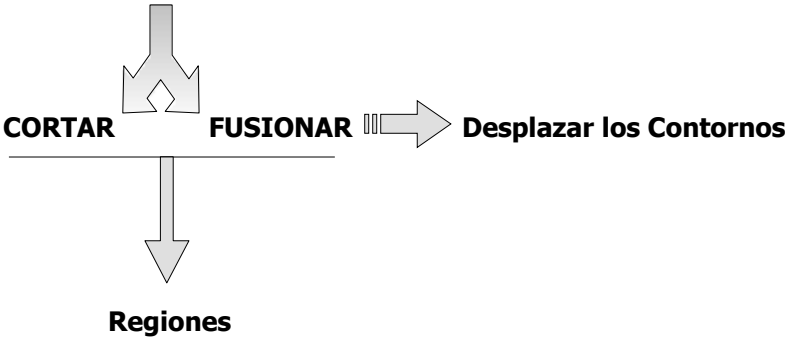


Fig.2.11.- Diagrama del Proceso Global de la Segmentación de la Imagen

2.5.1.-INTRODUCCIÓN.-

Estrategia:

- Iniciar una partición
- Definir un Criterio de Homogeneidad
- Optimizar (modificar) la Partición (mín. o máx. del Criterio)



Criterio de Homogeneidad:

1.-Determinista:

Textura: Varianza respecto a un modelo

$$C_T = \sum_n \sum_{i,j \in R_n} [x(i,j) - M_n(i,j)]^2$$

R_n : Región

i,j : Posición

Nivel de gris

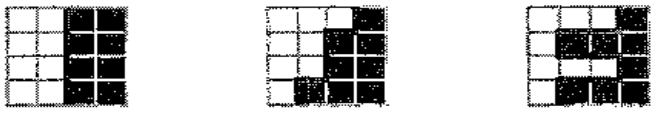
Modelo de la Región:
- Media
- Polinomio,...

Contorno: Minimizar la longitud para tener Contornos simples

C_C=Longitud Contornos

$C = \alpha C_T + (1-\alpha) C_T$ Criterio Global

Ejemplos de cálculo de la longitud de Contornos



L:4 L:6 L:10

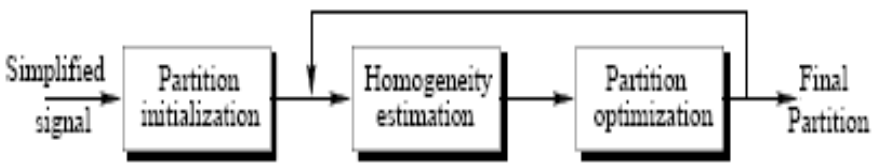
2.-Aleatorio:

Contorno: Estudio de Configuraciones locales

$$C = C_T . \prod e^{(Vc)}$$

Transiciones

Descripción de los Espacios de las características y de los algoritmos de decisión. El paso de Decisión en la segmentación basada en Homogeneidad busca la región actual donde los elementos son homogéneos con respecto al espacio de características.La mayoría de los procesos relevantes que implementan la Decisión mediante estas técnicas, son procesos iterativos. Podemos distinguir tres pasos importantes: *inicialización de la partición, estimación del Criterio de Homogeneidad y Optimización de la partición.*



a) .-Inicialización de la partición.El algoritmo es inicializado seleccionando una primera estimación de la partición.La primera estimación puede ser muy simple, como tener todos los elementos juntos en una única región o al revés, cada elemento asociado a una región diferente. Incluso esta primera estimación no tiene por qué ser una partición.Puede contener un conjunto de componentes que marquen la parte importante de las regiones (marcadores), y puede haber un gran número de elementos aún no asignados a ninguna región.El algoritmo iterativo construye la partición a partir del espacio de homogeneidad de características elegido, y de un algoritmo de optimización.

b).-Criterio de Homogeneidad.Según lo tratado ,las aplicaciones multimedia requieren el uso de características espaciales. Las características son todavía más necesarias en el caso de segmentación espacial.Para extraer los contornos espaciales de los objetos en una escena genérica, deben combinarse varias características básicas.

Un objeto es una entidad con el significado semántico que normalmente se asocia a un grupo de regiones conexas cuyos contornos están definidos en la imagen original.Este concepto,hace que utilicemos las características del color y de la textura en el proceso espacial de la segmentación. Además, las características de contornos se pueden introducir para evitar contornos ruidosos y obtener regiones basadas en la partición natural del objeto. Esta selección automática de regiones forman el objeto,confiando en las características de homogeneidad.
La información del movimiento,en gran parte, es aplicada desde regiones con movimiento homogéneo. Los escenarios, con movimiento de objetos en el primer plano y estáticos o quasi-estácticos en el fondo de la secuencia de vídeo, son utilizados con frecuencia. Al igual que en la Decisión basada en Transiciones, la estimación global del movimiento y compensación se puede utilizar para simplificar la imagen y obtener un estado quasi-estático de las secuencias del fondos del vídeo.

Aunque algunas técnicas se basan sólamente en la característica de movimiento, el funcionamiento de la segmentación mejora al combinar el movimiento con información de color y de textura.La Información de movimiento de la segmentación Directa, no genera transiciones exactas.Los contornos de los objetos están definidos de forma más exacta cuando se parte de la información de color.

La combinación de estas características espaciales viene dada por una estratégia paralela o jerárquica. La estimación mediante una *estrategia paralela* nos devuelve por un lado la segmentación espacial de movimiento, y por el otro la de color.Entonces ambas particiones se combinan.La e*strategia jerárquica* confía más en la combinación paso a paso.Comienza,

(1).-en un primer paso,toma una segmentación espacial color y

(2).-cambia los criterios para obtener una segmentación espacial
de movimiento en un segundo paso.

Algunos objetos no satisfacen la condición de presentar un movimiento homogeneo pero pueden verse como la unión de subobjetos con sus propios movimientos. Este es por ejemplo,el caso de regiones que corresponden al cuerpo del ser humano.La información de la profundidad es una útil característica para segmentar objetos complejos.Como antes, la profundidad puede ser estimada y combinada con las características anteriores en paralelo o de manera jerárquica.Para algunos objetos específicos es posible introducir en el proceso de la segmentación características semánticas.Las características semánticas se seleccionan después de analizar el comportamiento y las características del objeto.Un típico ejemplo incluye la segmentación de las partes de la cara.

Cuando el objeto que se sigue en una segmentación espacio-temporal es demasiado complejo, o sus características pueden variar en el tiempo, utiliza la interacción del usuario para definir o para actualizar el objeto.Téngase en cuenta, que para varias de las característica comentadas previamente, siempre es posible la ayuda automática mediante interacción del usuario, de esta manera, se mejorará el funcionamiento de la segmentación.

Podemos hablar de dos tipos de criterio de homogeneidad: **determinista y aleatoria**.

Para los ***Criterios Deterministas***:

$$C_T = \sum_n \sum_{i,j \in R_n} [x(i,j) - M_n(i,j)]^2$$

R_n : Región

i,j : Posición

donde el modelo $M_R(i,j)$ de la región R_n se ajusta a los dato x(i,j) y la distancia es calculada entre el modelo de los datos y los mismos datos.Típicamente, se utilizan los modelos polinómicos de la orden bajo y la norma es la distancia de Euclidea.

Para los ***Criterios Aleatorios***, se asume que la región es una realización de un campo aleatorio Vc, y usualmente se calcula una probabilidad 'a posteriori' de que los datos sean una muestra de este campo aleatorio.Teniendo en cuenta la independencia entre las regiones y entre los píxeles vecinos, la expresión de la probabilidad es la siguiente:

$$C = C_T . \prod_{\text{Transiciones}} e^{(Vc)}$$

Los modelos Gaussianos se utilizan con mucha frecuencia para caracterizar estos campos aleatorios.De acuerdo con la probabilidad estimada,se dice que la región es o no homogenea.

En gran número de casos, las implementaciones prácticas de estos dos tipos de criterio conducen a algoritmos muy similares.

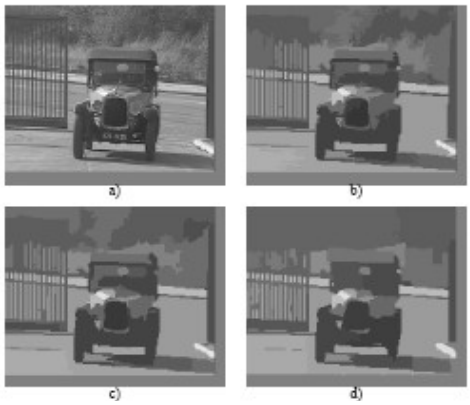


Fig.2.12.- Ejemplo de Criterio de Homogeneidad combinando textura con característica de color.

a) imagen original, b)peso bajo c)peso medio y d)peso alto.

c).-Optimización de la Partición.Las particiones son modificadas y actualizadas para alcanzar un grado óptimo en el sentido del Criterio de Homogeneidad.Las modificaciones parten de dos conceptos básicos: *la separación y la fusión.*

Dada una región, su bonad es juzgada en términos del Criterio de Homogeneidad seleccionado, y la región es separada,si esta división conduce a una mejor configuración.No todas las divisiones posibles de la región original son analizadas, sino un conjunto predefinido de divisiones.Normalmente, la región se divide en partes de igual área de soporte o en dos partes iguale separadas por una línea divisoria.La fusión se aplica a dos regiones cuando su unión en una única región conduce a una mejor partición en el sentido del Criterio de Homogeneidad.

En función del criterio de homogeneidad utilizado podemos obtener mejores o peores resultados de la fusión.Podemos ver un ejemplo de esto en la Fig.8,donde la misma imagen ha sido segmentada usando como criterio de Homogeneidad, una combinación de homogeneidad de textura y suavidad de contornos.En todos los casos, el número de regiones es el mismo.Una vez ponderada la disminución (aumento) de los contornos suavizados, las áreas texturadas aparecen segmentadas dentro de las regiones con particiones complejas (simples); ej. el bosque en el fondo de las fotos. Por otro lado, la segmentación de regiones altamente contrastadas produce particiones similares,independientemente del valor ponderado para los contornos suavizados; *ej. la puerta.*

Crecimiento de Regiones. Es el proceso básico para la fusión de regiones, se combinan basandose en el Criterio de Homogeneidad,hasta alcanzar una decisión final. El proceso puede ser inicializado definiendo un sistema de etiquetado que cubre parcialmente o completamente la partición,donde las regiones pueden estar formadas únicamente por un píxel. El análisis de todas estas posibles combinaciones puede resultar costoso para Criterios de Homogeneidad complejos. Debido a esto, es muy frecuente el uso de estrategias para reducir el número de fusiones analizadas.

El *Algoritmo Split & Merge* .La idea es dividir la imagen en regiones disjuntas, y luego ir dividiendo a su vez cada región en más regiones (split), hasta que se cumpla el criterio de homogeneidad. Luego empezar a fusionar estas regiones (merge) de manera que se siga cumpliendo el criterio para las fusiones.

El análisis de estos procesos ha sido llevado acabo desde el punto de vista de la aplicación y de las espacio de características complejas.A continuación se ofrecerá una explicación más detalla de los algoritmos de Decisión Iterativa mencionados en las líneas anteriores.

2.5.2.- ALGORITMO DEL CRECIMIENTO DE REGIONES

En primer lugar, se define en que consiste el Algoritmo de Crecimiento de Regiones, de forma que podamos conocer los criterios que se han tomado de cara a su implementación.

Es un algoritmo basado en colas jerárquicas, donde se introducen los diferentes píxeles previamente a su asignación a una determinada región siguiendo un determinado criterio. Habrá tantas colas como número de colores (o niveles de gris) tenga la imagen original, y se corresponderá con diferentes prioridades a la hora de asignar los píxeles que contengan regiones determinadas.El algoritmo de Crecimiento de Regiones puede dividirse en dos etapas:fase de *inizialización* del algoritmo y *evolución* del mismo.

INIZIALIZACIÓN. El proceso parte de dos imágenes.La primera es la imagen original que pretendemos segmentar y la segunda las zonas seguras. El primer paso a realizar es la inicialización del sistema. Para ello se parte de la imagen correspondiente a las zonas seguras. Se introducen los píxeles (concretamente sus coordenadas dentro de la imagen original) pertenecientes a estas zonas en la cola de máxima prioridad,entendiendo como cola de máxima prioridad aquella cola a la que primero se accederá a la hora de extraer píxeles.

EVOLUCIÓN. Una vez colocados en la cola de máxima prioridad todos los píxeles de las zonas seguras, se inicia la evolución del algoritmo. Para ello, se comienza extrayendo el primer píxel que se encuentre en la cola de máxima prioridad. Debido a la fase de inicialización, esta cola seguro que estará vacía.

Se recorren las colas hasta encontrar una que no esté vacía, de la cual se extraerá el primer píxel. El tratamiento de este píxel será el siguiente:

-Sí el píxel extraído tiene un único vecino que ya esté asignado a una región(siempre tendrá al menos uno), será asignado a dicha región. A continuación, tal como se ha explicado anteriormente, se hallarán aquellos vecinos que pertenezcan a la zona de incertidumbre, y si los hay, se calcularán sus distancias respecto a la región asignada al píxel extraído, y serán introducidos en las colas que les corresponda.

-Si el píxel extraído tiene varios vecinos que ya estén asignados a una determinada región, se calculará a cuál de estas regiones tiene menor distancia, y será asignado a ella. A continuación se procederá con sus vecinos, si los tiene, de la misma forma que en el caso anterior.

El Algoritmo seguirá evolucionando de forma análoga hasta que todas las colas estén vacías.Existe la posibilidad de que analizando píxeles de una cola de prioridad N, encolemos uno en la cola de prioridad mayor, con lo cual deberemos retroceder hasta ésta para seguir el proceso. Vaciadas todas las colas se habrá eliminado la zona de incertidumbre, y por tanto, se habrá obtenido la imagen segmentada.

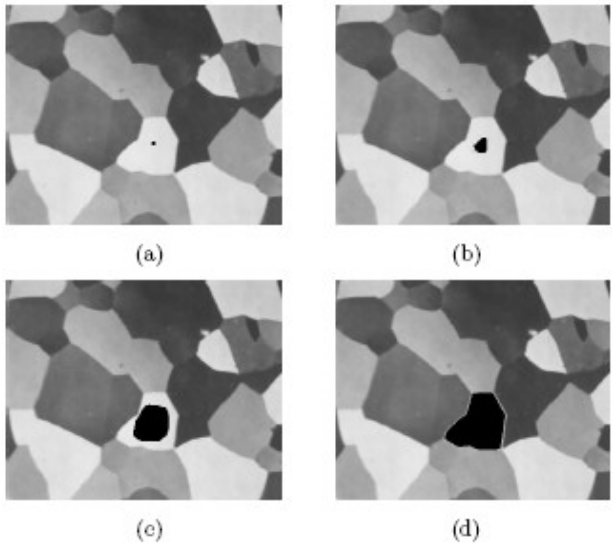


Fig.2.13.- (a) Imagen original con región marcada mediante pto de selección, (b)-(d)Proceso de crecimiento de la región marcada.

2.5.3.-ALGORITMO DE SPLIT & MERGE.-

El algoritmo de Split&Merge, como su nombre indica, engloba dos procesos característicos, uno de división y el otro de fusión.

SPLIT.- A partir de la imagen original, que es considerada como el primer gran bloque, se proced a aplicar el split, es decir, se le aplica un determinado criterio de homogeneidad por el que, si dicho bloque no lo cumple, se divide geométricamente en cuatro partes, en cuatro bloques, que a continuación serán analizados por el mismo criterio. Dicho proceso se repite hsta que todos los bloques cumplan el criterio de división.

Para definir el criterio que se ha tomado en la implementación de la aplicación en Matlab 6.5 (que es el software utilizado para llevar acabo el proyecto) se parte de la diferencia entre los nivels de gris máximos y mínimos de los píxeles del bloque analizado y un umbral introducido por el usuario. El criterio se cumple, y por lo tanto no se produce un nuevo split, siempre y cuando la diferencia entre los valores de los píxeles citados sea inferior al umbral introducido.

MERGE.- Una vez finalizada la fase de split, se procede ha realizar la de Merge, fase que une los bloques creados.En este caso, se ha de comprobar si dos bloques consecutivos,adyacentes, es decir, que estén en contacto, cumplen el criterio anterior. Si es así, dichos bloques quedarán fusionados, formando un único bloque. Dicho proceso se repite hasta llegar a la idempotencia, es decir, el momento en el cual ya no pueden fusionarse más bloques.

El criterio adoptado en ete proceso es igual que el aplicado en el anterior, con el apunte adicional que el umbral de decisión puede ser variado entre la etapa de split y la de merge. Por lo general se suele tomar un umbral bastante estricto al realizar el Split, suavizándolo a continuación en el Merge.

Las principales *ventajas* de este algoritmo son, en primer lugar, su sencillez y simplicidad, tanto el algoritmo en sí ,como la partición inicial desde la que se inicia el mismo (imagen original como primer gran bloque a tratar). Por otro lado, se tiene una visión global, y no local, de como va evolucionando el algoritmo.

El principal *inconveniente*, es sin duda,la forma que tiene de segmentar, de dividir los bloques, pues es totalmente geométrica,obteniendose contornos no muy naturales.

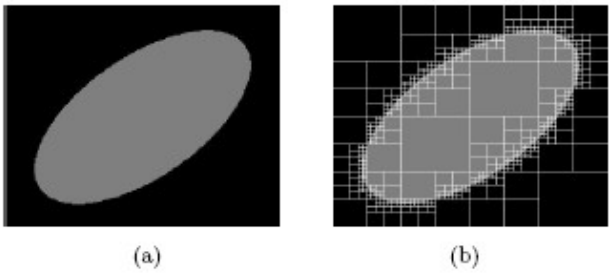


Fig.2.14.-(a)Imagen sintética, (b)Imagen dividida y fusionada

2.5.4.-SEGMENTACIÓN POR WATERSHED.-

En el campo del procesado de imagen, y más concretamente en el de la morfología matemática, las imágenes de niveles de gris son habitualmente consideradas como relieves topográficos. En la representación topográfica de dichas imágenes la altura de cada punto corresponde con el nivel de píxeles correspondiente. Esta representación es muy adecuada para poder percibir mejor el efecto de una deteminada transformación sobre una imagen. Por ejemplo, que una apertura elimine algunos picos y líneas de cresta,mientras que el cierre tiende a rellenar valles o pequeñas llanuras.

La técnica de Watersehed constituye una de las más poderosas herramienta de segmentación aportada por la morfología matemática.

APLICACIÓN DEL WATERSHED EN LA SEGMENTACIÓN.

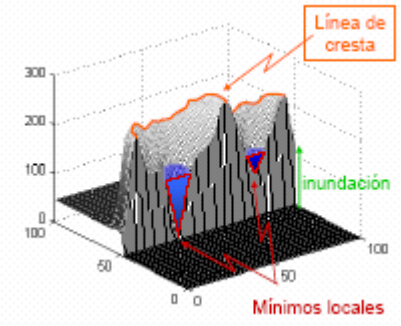
Análogamente a otras técnicas de segmentación, el objetivo de la técnica del Watershed es dividir en regiones la imagen de nivel de grises analizada. Generalmente una de ellas se corresponde con el fondo de la imagen y el resto con los objetos o regiones que se pretende extraer. El objetivo último de esta técnica es determinar los contornos que definen dichos objetos. En este punto el problema es definir qué es contorno y que no lo es.

Algunas de las técnicas se basan en el estudio de los cruces por cero de la segunda derivada o en detector de determinados tipos de perfiles. En el campo de la morfología matemática, son otro tipo de aproximaciones las que habitualmente se utilizan.

El punto de partida es considerar que los contornos de una imagen se corresponden con las líneas donde el nivel de gris varía más rápidamente que en un determinado entorno vecino.Se define la imagen grad(I), la cual está formada por píxeles cuyo valor es el módulo del gradiente en dicho punto (conocida como imagen gradiente). De esta manera, se considera que los contornos de la imagen original se corresponden con las líneas de cresta de la imagen gradiente.

Algoritmo de Watershed.-El concepto de watershed se basa en visualizar una imagen en 3 dimensiones (3D): dos coordenadas espaciales vs. niveles de gris.
En esta interpretación 'topográfica', consideramos tres tipos de puntos:

1. puntos que corresponden a mínimos locales.
2. puntos en los que,si se coloca una gota de agua, esta cae con certeza en un único mínimo.
3. puntos en los que el agua caería con igual probabilidad en más de uno de estos mínimos.



Para un mínimo local particular, el conjunto de puntos que satisfacen la condición (2) se llama ***catchment basin o watershed o cuenca*** de este mínimo.Los que satisfacen la condición (3) forman líneas de cresta en la superficie topográfica y son llamadas ***líneas divisorias o líneas de watershed o líneas de cresta.***El objetivo principal de los algoritmos de segmentación es alcanzar estas líneas divisorias o de cresta.

Fig2.15.- Línea de cresta

La idea básica es la siguiente: supongamos que se hace un pequeño agujero en cada mínimo local, y que todo el relieve topográfico es inundado desde abajo, dejando que el agua entre a velocidad constante.

El agua va subiendo e inundando las cuencas. Cuando el agua de dos cuencas está a punto de juntarse, se construye un dique (dam) para evitar la fusión.

La inundación continúa, y llega a un punto en que solo se ve la parte de arriba de los diques por arriba de la línea de agua.
Las líneas de watershed forman un camino conexo, dando por lo tanto bordes continuos entre las regiones.

Por otro lado, el uso de la imagen gradiente ,para aplicar directamente sobre ella la técnica de watershed, no constituye un

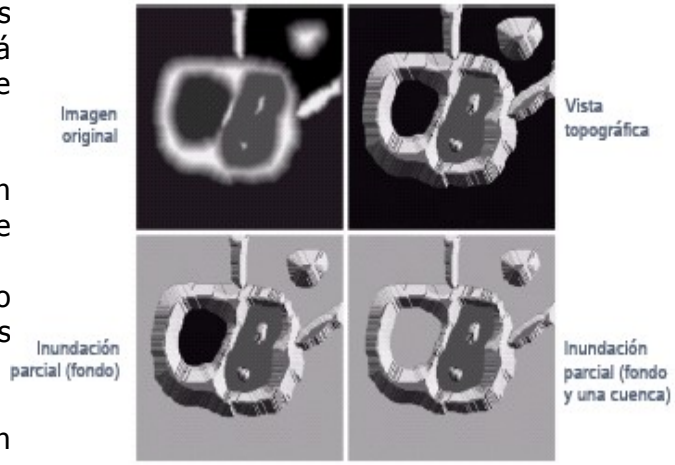
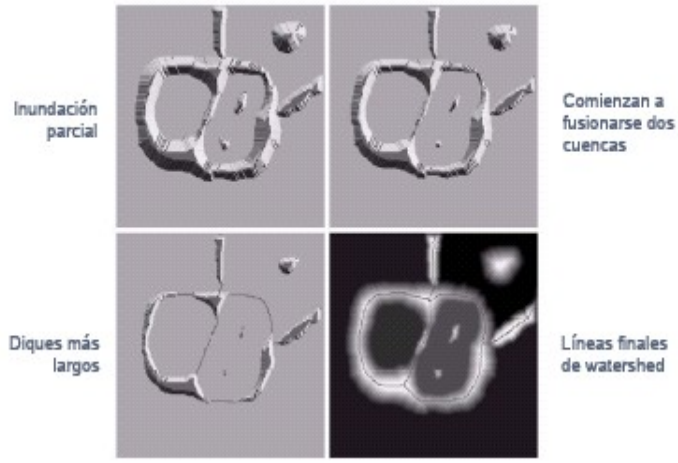


Fig.2.16.-Watershed fase I.



buen método de segmentación. Pues habitualmente el uso directo de la imagen gradiente para detectar las líneas de contorno produce el efecto conocido con el nombre de *sobresegmentación.* Este efecto provoca que los verdaderos contornos queden enmascarados por una infinidad de contornos falsos o irrelevantes. Esto es así aunque se haya tomado la precaución de filtrar previamente la imagen original o su correspondiente imagen gradiente.

Fig.2.17.-Watershed fase II.

Habitualmente, la *sobresegmentación* es debida a la presencia de ruido. Pero también los efectos de las texturas o de objetos presentes poco relevantes que no se desean segmentar provocan la aparición de falsos contornos asociados a multitud de valles o zonas de depresión que no se corresponden con objetos o regiones.

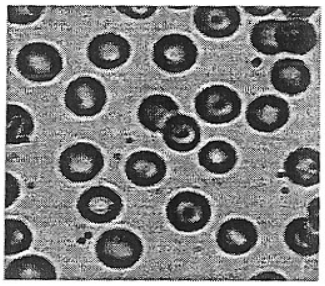
Para solucionar esta *sobresegmentación* existen dos posibilidades:

1.-Eliminar los contornos irrelevantes una vez realizado el watershed.

2.-Modificar la imagen gradiente de tal forma que las regiones de depresión o valles se correspondan únicamente con los objetos deseados.

1.-Eliminar los contornos irrelevantes una vez realizado el watershed:

Para implementarla, debemos considerar la imagen obtenida como un conjunto de contornos de los cuales se debe eliminar los que no sean representativos de regiones u objetos de la imagen.O considerar la imagen como un conjunto de regiones adyacentse que se pueden unir siguiendo un determinado criterio, para que finalmente sólo se tengan las regiones que se correspondan con los objetos que se pretendía detectar. Esta segunda posibilidad se englobaría en el conjunto de algoritmos de Crecimiento de Regiones, analizado anteriormente.



Original: sangre

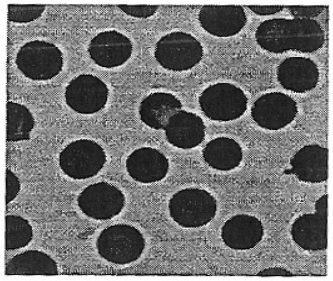
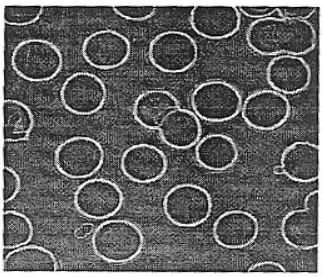
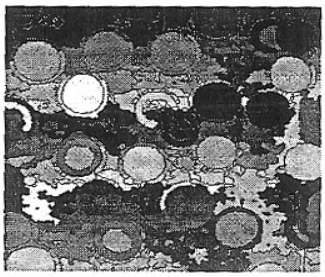


Imagen simplificada
apertura con reconstrucción
cierre por reconstrucción: 10x10

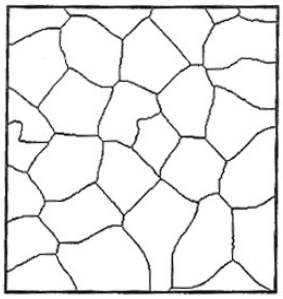


gradiente



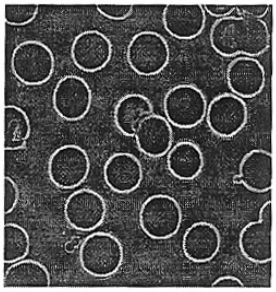
Segmentación

Marcadores de los objetos:
Binarización.

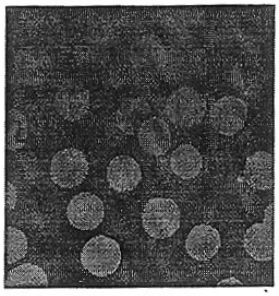


Marcador del fondo:
watershed de los marcadores de los objetos

Marcadores



Gradiente simplificado:
limitación de los mínimos



Segmentación final

la imagen gradiente y evitar el efecto de sobresgmentación.

Una vez extraídos los marcadores, una transformación morfológica basada en operacions geodésicas permite:

1.-Imponer marcadores como mínimos de la imagen gradiente.

2.-Suprimir todos los demás mínimos del gradiente, los irrelevantes, rellenando los correspondientes valles.

3.-Preservar las líneas de cresta más importantes de la imagen gradiente localizadas entre los marcadores.

Esta transformación es denominada variación de la homotopía del gradiente, o simplemente, modificación del gradiente. El cálculo de las líneas de cresta o watershed de la imagen graciente modificada aporta los resultados esperados: el marcador del fondo recompone el mismo, mientras que el rsto de los contornos obtenidos se corresponden con los límites de los objetos que se deseaba detectar.

CAPÍTULO II
FUNCIONAMIENTO DEL EDITOR DE IMÁGENES

3.1.-INTRODUCCIÓN AL ENTORNO MATLAB.-

3.1.1.-Imágenes en Matlab.Tipos de Datos.

La estructura básica de datos en Matlab es una matriz, un conjunto ordenado de elementos reales.Este tipo es naturalmente adecuado para la representación de imágenes, vistas éstas como un conjunto de píxeles con información de color o intensidad de gris.

Matlab almacena la mayoría de tipos de imágenes que soporta como matrices bidimensionales, cada elemento de la cual hace referencia a un píxel de la imagen representada (píxel es un anglicismo qu eproviene de la abreviación *picture element*, y normalmente representa un punto de luz en el monitor).Por ejemplo, una imagen compuesta por un conjunto de 144 filas y 176 columnas de píxeles (que es el tipo de imágenes que se han utilizado para las pruebas del editor) puede ser almacenada en Matlab en una matriz bidimensional de 144x176 o tridimensional si es imagen color de144x176x3.

Esta posibilidad de almacenamiento supone que trabajar con imágenes en Matlab sea similar a trabajar con cualquier otro tipo de dato matricial.Por lo que sería posible seleccionar un determinado píxel de la imagen mediante el formato típico de acceso a un elemento de una matriz:

I(123,56) *representación matricial de la imagen*

este comando retornaría el valor del píxel de la fila 123,columna 56 de la Imagen I.

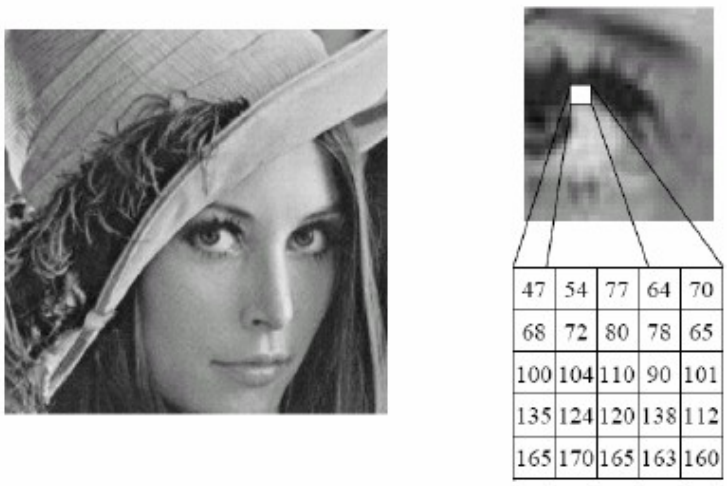


Fig.3.1.-Imagen original,representación matricial de la imagen

-Tipos de Datos.

Por defecto, Matlab almacena la mayoría de datos en matrices de clase double. Los datos de estas matrices son almacenados como números de coma flotante de doble precisión (64 bits). Todas las funciones y capacidades de Matlab trabajan con este tipo de matrices.

Pero, para el procesado de imagen, este tipo de representación no es siempre ideal. El número de píxeles de una imagen puede ser grande, por ejemplo, una imagen de 1000x1000 píxeles tendría un millón de píxeles. Esta manera de representar la imagen necesitaría alrededor de 8 megas de memoria.

Con el objetivo de reducir estas necesidades de memoria, Matlab proporciona,en una de las versiones recientes de hace unos años, la clase de matriz uint8. Los datos en este tipo de matrices son almacenados como enteros de 8 bits, con su correspondiente ahorro de memoria.

Debido a la coexistencia de estos dos tipos de formatos, Matlab, y en concreto el Toolbox de Procesado de Imagen utilizan diferentes pautas para interpretar loa valores en estas matrices. El siguiente apartado de esta introducción analiza como el Toolbox interpreta las diferentes clases de datos así como qué tipo de operaciones son permitidas en función de la clase de matriz.

3.1.2.-Tipos de imágenes en el Toolbox.

El Toolbox de Procesado de Imagen, soporta cuatro tipos básicos de imágenes:

-Imágenes indexadas.

-Imágenes de intensidad de gris.
-Imágenes binarias.
-Imágenes RGB.

Este apartado analiza cómo Matlab y el Toolbox de Procesado de Imagen representan cada uno de estos tipos de imágenes.

Imágenes indexadas

Una imagen indexada consiste en dos matrices, una matriz imagen y una *colormap.* La matriz de *colormap* es un conjunto ordenado de valores que representan los colores de la imagen. Para cada píxel de la imagen, la matriz imagen contiene un valor que es un índice referente al *colormap.*

El *colormap* es una matriz *mx3* de clase *double*. Cada fila de la matriz *colormap* especifica los valores de rojo, verde y azul (RGB) de un determinado color:

color = [R G B];

R, G y B son valores reales dentro del margen 0 (negro) a 1.0 (máxima intensidad).
La siguiente figura ilustra la estructura de una imagen indexada. Los píxeles en la matriz de la imagen son representados por enteros, los cuales son punteros o índeces a los valores de un color almacenado en la matriz colormap. Si la iamgen indexada fuera una imagen en niveles de gris y no en color todos los valores de R, G y B tendrían el mismo nivel de intensidad por lo que los i,j de la matriz colormap serían iguales dentro de la misma fila.

La relación entre los valores de la matriz imagen y su correspondencia con los colores en la matriz colormap depende de si la matriz imagen es de clase double o uint8. Si la matriz imagen es de clase double, el valor 1 apunta a la primera fila del colormap,el valor 2 a la segunda, y así progresivamente. Si es de clase uint8, se produce un desplazamiento de tal manera que el valor 0 apunta a la primera fila del colormap, el valor 1 a la segunda, etc...

Por ello en la figura se observa que la flecha del valor 2 apunta a la posición de la segunda fila en la matriz colormap.

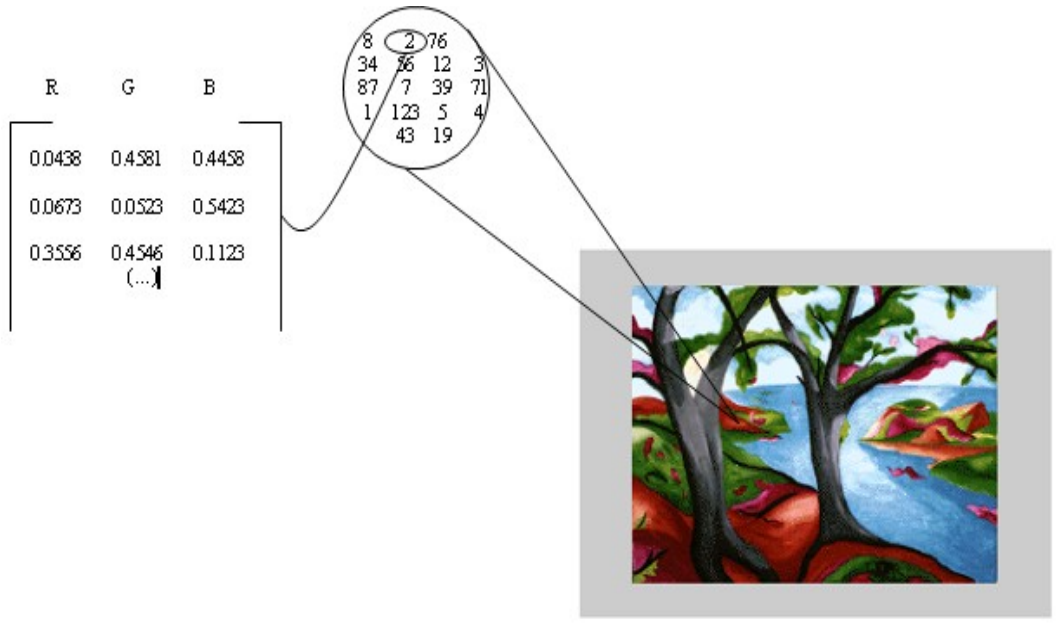


Fig.3.2.-Representación de imágenes indexadas: imagen indexada original, parcial de la matriz imagen y matriz colormap.

Imagenes de Intensidad

Matlab almacena una imagen de intensidad ocmo una matriz simple, en la cual, cada elemento corresponde con un píxel de la imagen. La matriz puede ser de clase *double*, caso en el cual dicha matriz contiene valores en el margen [0,1], ode clase *uint8*, en cuyo caso el margen es [0,255].

Fig.3.3.-Imagen nivel de gris

Imagenes Binarias

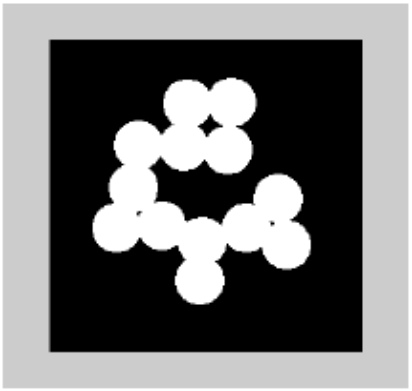


Fig.3.4.- Imagen Binaria

En una imagen binaria, cada píxel puede tomar uno de dos posibles valores discretos. Esencialmente, estos dos valores corresponden a *on* o *off.*Una imagen binaria es almacenada como una matriz bidimensional de ceros *(píxeles off)* y de unos *(píxeles on)*.

Una imagen binaria puede ser considerada una clase especial de imagen de intensidad, conteniendo sólo los valores correspondientes al negro y al blanco.Una imagen binaria puede ser almacenada en una matriz de clase *double* o clase *uint8.*De todas formas, es preferible la clase uint8 debido a su menor utilización de memoria.

En el Toolbox de Procesado de Imagen cada función que retorna una imagen binaria como una matriz lógica de tipo uint8. El Toolbox utiliza la etiqueta lógica para significar que el rango es [0,1]. Si la etiqueta lógica está *off* se asume que el rango es [0,255].

Imágenes RGB

Como en una imagen indexada, una imagen RGB representa el color de cada píxel como un conjunto de tres valores que contienen la intensida de rojo, verde y azul para conformar el color. Pero a diferencia de las indexadas, estas intensidades son almacenadas directamente en matrices imagen, no en una matriz *colormap.*

En matlab, las componentes de color rojo, verde y azul de la imagen RGB reisden en una matriz *mxnx3,* donde *m* y *n* son el número de filas y de columnas de la imagen, y la tercera componente representa la dimensión, indicando que consta en este caso de 3 planos, conteniendo los colores de rojo, verde y azul de cada píxel.

Por ejemplo, para determinar el color del píxel (112,56) se ha de buscar en la tripleta RGB almacenada en (112,56,1:3). Suponiendo que el valor contenido en (112,56,1) fuera 0.4561, el de (112,56,2) fuera 0.1234 y por último, el de (112,56,3) fuera 0.0378, el color del píxel sería:

0.4561 0.1234 0.0378

Una matriz imagen que almacena un tipo RGB puede ser de tipo *double*,en cuyo caso el margen de los valores es [0,1], o de tipo *uint8*, y entonces el margen corresponde a [0,255].

Fig.3.5.- Representación de Imágenes RGB:imagen RGB y representación matricial

Rutinas de lectura y escritura.

Es posible utilizar la función de Matlab ***imread*** para leer imágenes desde ficheros con el formato gráfico BMP, HDF, JPEG, PCX,TIFF y XWD,además del formato propio MAT.
Para escribir imágnes en ficheros, Matlab aporta la función ***imwrite***, que soporta los mismos formatos que *imread.*

Convertir imágenes a otros tipos.

Para realizar ciertas operaciones,ayuda el hecho de cambiar el tipo de imagen sobre la cual se está trabajandtro.Por ejemplo,para filtrar una imgen de color que está almacenada en una imagen indexada,primero se ha de transformar ésta en un aimagen de tipo RGB.Al aplicar el filtro sobre la imagen RGB, Matlab filtra las intensidades de los valores de la imagen, que es lo que se pretende.Pero si se filtra la iamgen en formato indexada, Matlab simplemente aplica el filtro a los índices de la matriz imagen, y el resultado no tiene ningún sentido.

El Toolbox de Procesado de Imagen provee de numerosas funciones,(que han sido utilizadas en nuesto sofftware) que permiten las conversiones entre diferentes tipos de imágenes.

3.2.-DESCRIPCIÓN E IMPLEMENTACIÓN DE LA INTERFACE.-

3.2.1.-Descripción General.-

El Objetivo es realizar un conjunto de funciones en Matlab que permitan:

- Abrir una imagen seleccionándola, si el usuario quiere, del directorio de imagenes que se adjunta a este software, imagenes de disco originales en escala de gris o color real , así como sus respectivas componentes color y luminancia y particiones de las imagenes originales.

La imagen podrá ser en escala de grises,indexada o en color real, en función del algoritmo

utilizado, se harán las conversiones pertinentes. Además, el usuario tiene la posibilidad como funciones extras de obtener la conversión a binaria o escala de grises de la entrada de la imagen original.

- Como herramientas de edicción, se han intentado implementar funciones como: deshacer, rehacer, recortar o zoom. Actualmente, a pesar de la existencia del código de implementación de las misma, no están activas. Son funciones, pendientes de revisar y corregir en un futuro.

- A nivel de particiones, tenemos la posibilidad de:
  -Cargar una partición de disco
  -o generar una nueva a partir de la segmentación de la imagen original.

La creación de una partición nueva, se presenta a través de tres algoritmos diferentes de segmentación, explicados ampliamente en el capítulo anterior. En primer lugar, se trata el algoritmo de K-means, una de las heurísticas comúnmente utilizadas para resolver el problema de clustering.Un segundo algoritmo implementado es el de Watershed, poderosa herramienta de segmentación aportada por la Morfología matemática,para ello hemos usado una de las funciones incluidas en el Toolbox de procesado de imagen que ofrece el entorno Matlab, denominada '*Watershed*'.

Como parámetro de entrada, el usuario podrá seleccionar la conectividad,por defecto la función de Wathershed utiliza conectividad 8-conn para imagenes de 2D *(imágenes en escala de grises)* y conectividad 26-conn para imágenes de 3D *(imágenes color)*.

Si hacemos,por ejemplo, un help en la línea de comandos de matlab de la función de watershed obtendremos una breve descripción, de los parámetros requeridos y de su funcionamiento:

WATERSHED Find image watershed regions.
L = WATERSHED(A) computes a label matrix identifying the watershed regions of the input matrix A. A can have any dimension. The elements of L are integer values greater than or equal to 0. The elements labeled 0 do not belong to a unique watershed region. These are called "watershed pixels." The elements labeled 1 belong to the first watershed region, the elements labeled 2 belong to the second watershed region, and so on.

By default, WATERSHED uses 8-connected neighborhoods for 2-D inputs and 26-connected neighborhoods for 3-D inputs. For higher dimensions, WATERSHED uses the connectivity given by CONNDEF(NDIMS(A),'maximal').

L = WATERSHED(A,CONN) computes the watershed transform using the specified connectivity. CONN may have the following scalar values:

4 two-dimensional four-connected neighborhood

8 two-dimensional eight-connected neighborhood
6 three-dimensional six-connected neighborhood
18 three-dimensional 18-connected neighborhood
26 three-dimensional 26-connected neighborhood

Connectivity may be defined in a more general way for any dimension by using for CONN a 3-by-3-by- ... -by-3 matrix of 0s and 1s. The 1-valued elements define neighborhood locations relative to the center element of CONN. If specified this way, CONN must be symmetric about its center.

Class Support

A can a numeric or logical array of any dimension, and it must be nonsparse. The output array L is double.

Otro de los algoritmos propuestos, pero que no está activo en la interficie, es el Algoritmo aportado por *Luís Garrido y Philippe Salembier* al software *Soft-Image* del Grupo de Imagen del *Departamento de Teoría de la Señal y Comunicaciones* de la Upc para la Titulación de Ingeniería en Telecomunicaciones.Es una librería programada en C,en el apéndice A de este documento se ofrece el código Mex, necesario para la implementación de este algoritmo en la interface creada en Matlab.La función Mex no está compilada ni añadida en la guide principal de este Editor, es una propuesta para mejoras futuras.

Precisamente, las particiones de disco que se adjuntan en la carpeta *image* de este proyecto, son generadas desde este algoritmo de segmentación, programado en C.

Otras funciones de edicción que podemos aplicar a la partición, son:

- Puesto que la partición es una imagen en niveles de gris segmentada y posteriormente etiquetada, podemos conocer el valor de su etiqueta clicando previamente con el ratón en aquella región de interés.

- A su vez podemos cambiar el valor original de esa etiqueta por otro de nuestro interés.

- Y posteriormente, hacer un cambio simultáneo de etiquetas con el fin de fusionar varias regiones en una, bajo nuestro Criterio de Homogeneidad.

- Podemos visualizar la partión,que originalmente es en niveles de gris, en color, con valores comprendidos entre [0,255] para imágenes uint8 y entre [0,1] para imágenes double, mediante una función de aleatoriedad, que utiliza la función matemática de random(), para devolver un valor aleatorio de cada una de las regiones etiquetadas de la partición, de manera que cada vez que presionas el botón de *random* en la interface, obtendrás un color diferente para cada región.

- El *ítem Properties* del *menú Partition* retorna las propiedades asociadas a la etiqueta de la partición seleccionada por el usuario.La función *regionprops* de Matlab es la que nos permite obtener esta información.

- Por último el *ítem Save* del *menú Partition* te da la posibilidad de guardar la partición

en disco, tanto la partición color como la de niveles de gris.

El proceso de composición de dos o más imágenes requiere de mucho control sobre qué zonas de cada una de ellas serán usadas, así como tener en cuenta el grado de transparencia. Una máscara es una imagen que puede ser manipulada como si se tratara de una imagen cualquiera, lo que la diferencia del resto es su propósito, se utiliza durante la composición cuando sólo se quiere incluir una parte de la imagen en el resultado.

Puesto que el objetivo final de software es la reconstrucción de la imágen original, utilizaremos esta máscara como multiplicando de la imagen original para obtener como resultado la imagen reconstruída. Siguiendo el orden de menús del guide, nos encontraremos el item para la creación de esta máscara, clicando en la partición obtendremos los puntos de interés para su creación.

Finalmente, pasaremos a la fase de reconstrucción de la imagen, partiendo de la máscara o de su invertida.

3.2.2.-Software GUIDSegmentation (Manual del Usuario).-

3.2.2.1.-Introducción.-

Esta sección se ha escrito como manual de usuario para la aplicación guide, cubre todas las opciones posibles que el usuario puede interactivar, explicando todos los parámetros que pueden ser introducidos para las diferentes opciones, así como todos los mensajes emergentes y botones.Incluye comentarios sobre métodos de programación para aclarar algunos puntos de interés.

3.2.2.2.-Funciones de Utilidad.-

Matlab permite desarrollar programas con el aspecto típico de las aplicaciones de Windows.Para todos los controles, se utilizará la función uicontrol, que permite desarrollar dichos controles.
La forma general del comando ***uicontrol*** es la siguiente:

```
id_control = uicontrol(id_parent,...
                       'Propiedad1',valor1,...
                       'Propiedad2',valor2,...
                        ... (otras propiedades)
                       'callback','sentencias')
```

Existen ocho tipos de controles diferentes:pushbuttons,checkbox, radio buttons, sliders,

pop-up menus,editable textboxes, static textboxes y frames.La utilización de cada uno de ellos vendrá en función de sus características y aplicaciones.

3.2.2.3.-Ventana Princial (Main windows).-

La Figura 3.6.- es una impresión de pantalla de la ventana principal de la aplicación. Debido a la posible utilización del software en un futuro con alumnos que puedan proceder de otros países, se decidió la terminología de la aplicación en lengua inglesa, por ser un lenguaje comúnmente conocido por la mayoría de los estudiantes.

Interfície Gráfica Guide: GUIDSegmentation

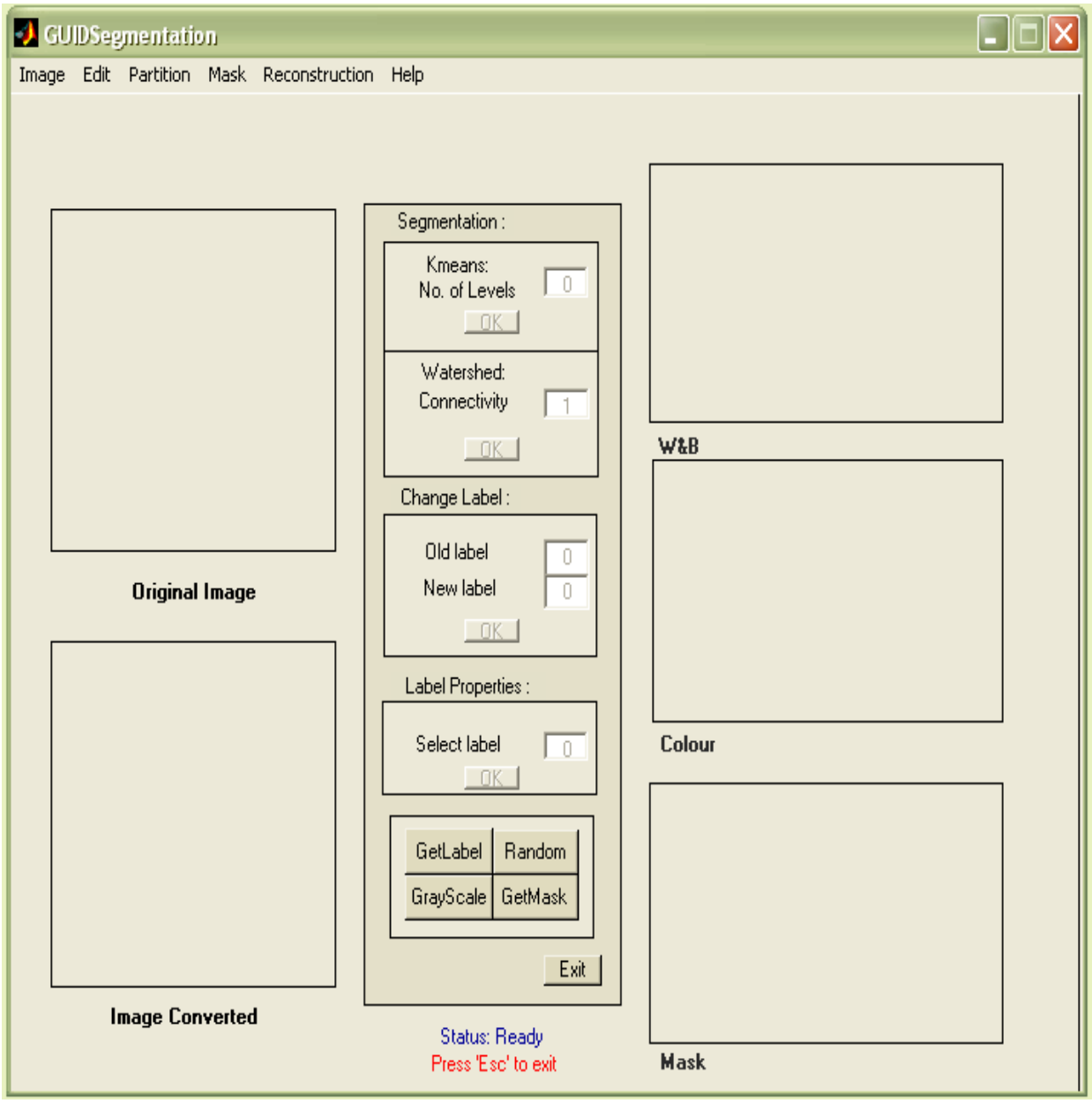


Fig.3.6.-Interficie gráfica Guide, *main window*

Main Menu

La barra de herramientas de la ventana principal, se desglosa en 6 menús,cada uno de ellos con sus respectivos submenús, estas son las opciones:

Menú Image:

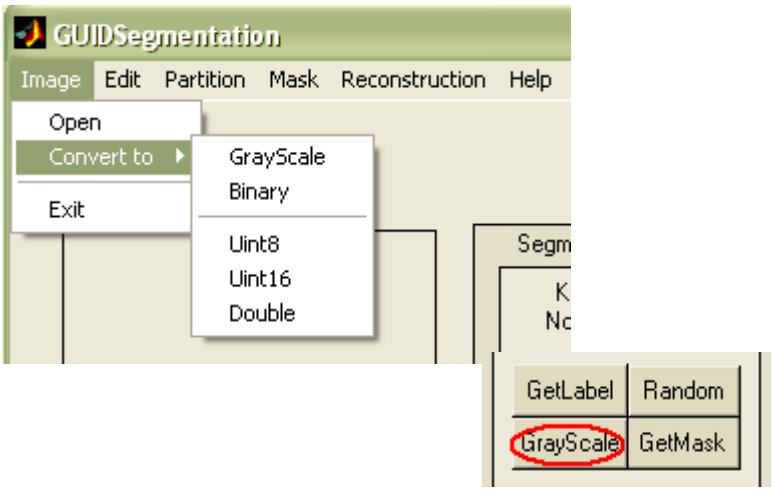


Fig.3.7.-Conver to>item GrayScale
>Botón GrayScale

a).-Item Open:

Al pulsar sobre el ítem se realizarán las siguientes funciones:
-se abrirá un diálogo con los directorios correspondientes a las carpetas donde se encuentras imágenes con las extensiones programadas en el código fuente:

```
[filename, pathname]=uigetfile...
('*.jpeg;*.jpg;*.tiff;*.gif;*.bmp;*.png;*.hdf;*.pcx;*.xwd;...
  *.ico;*.cur;*.ras;*.pbm;*.pgm;*.ppm;', 'Open input image');

if filename ~= 0
%Displaying the original image
        file = [pathname filename];
         originalimg = imread(file);
```

El usuario podrá elegir,entre todos estos formatos de imágenes.
Como se muestra en la figura siguientes:

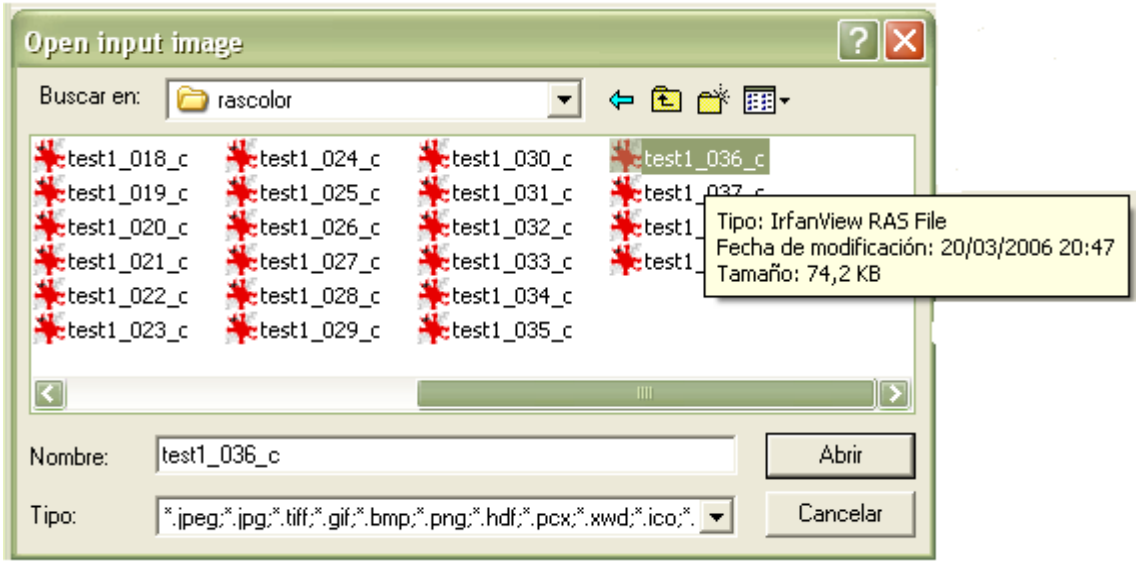


Fig.3.8.-Diálogo para abrir archivo *'imagen original'*

b).-Item Convert to:

El algoritmo permite seleccionar para la imagen original, abierta previamente, el tipo de imagen con la que se desea trabajar,haciendo una conversión de la imagen original al tipo de imagen seleccionada por el usuario, a su vez es posible elegir de igual forma que antes el tipo de dato de la imagen:*uint o double*.Como se muestra en la *Fig.3.7*.- al final de la página anterior.

Para la función de conversión en niveles de gris además del item podemos hacer la llamada desde el botón '*GrayScale*' como podemos apreciar en la *Fig.3.7* en la parte inferior derecha del gráfico.

Para las conversiones a los diferentes tipos datos o imagen hemos utilizado herramientas pertenecientes al Toolbox de procesado de Imagen de Matlab. Los formatos para las difentes funciones son los siguientes:

GrayScale:

Esta función ***rgb2gray*** convierte imágenes RGB o colormap en escala de grises.

I = RGB2GRAY(RGB) *convierte imágenes de color verdadero RGB a imágenes en escala de grises y el resultado lo guarda en la variable I.*

NEWMAP = RGB2GRAY(MAP)*, esta función también te da la posibilidad de devolverte un colormap en escala de grises equivalente al valor de entrada MAP.*

RGB *es la imágen de entrada.*
Map *el colormap de entrada.*

Binary:
Esta función ***im2bw*** convierte imágenes RGB, en escala de grises o indexadas a imágenes binarias tomándo un valor de umbral.

BW = IM2BW(I,LEVEL) la imagen es convertida a blanco y negro, utilizando un valor de umbral automático.

BW = IM2BW(X,MAP,LEVEL) convierte imágenes indexadas con un colormap determinado a imágenes binarias, en blanco y negro.

X,I *son imágenes de entrada.*
Level *es el valor de umbral.*

Las herramientas para conversiones entre tipos de datos son:

UINT8 *convierte los datos de la imagen de entrada a enteros de 8-bit sin signo.*
UINT16 *convierte los datos de la imagen de entrada a enteros de 16-bit sin signo.*

I = UINT8(X) *convierte los elementos de la matriz a enteros de 8-bits sin signo.La matriz (X) puede ser un objeto numérico de tipo Double.Los valores UINT8 van desde 0 a 255.*

I = UINT16(X) *convierte los elementos de la matriz a enteros de 16-bits sin signo.La matriz (X) puede ser un objeto numérico de tipo Double.Los valores UINT16 van desde 0 a 65535.*

X, *es la matriz de entrada.*

DOUBLE *convierte los datos a double precisión, por defecto en matlab los datos se crean en este tipo de precisión.*

I=DOUBLE(X) *devuelve datos de tipo double para los elementos de la matriz X. If X ya es una matriz de elementos double, la función no tiene efectos sobre ella.*

c).-Item Exit

Si el usuario quiere salir de la aplicación debe pulsar sobre este item y aparecerá en pantalla un diálogo en el que se les preguntará a modo de reconfirmación:' *si están seguros en salir de la aplicación*'.Como se muestra a continuación:

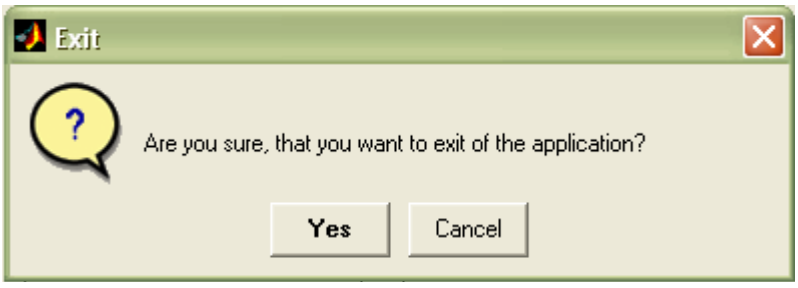


Fig.3.9.-diálogo de salida

Menú Edit:

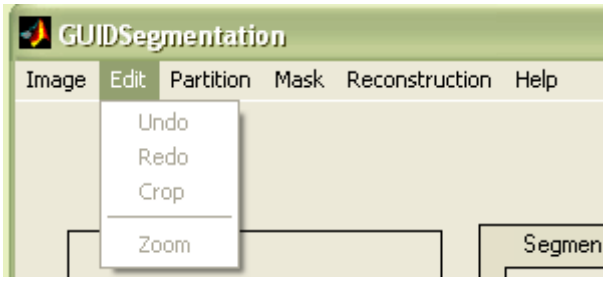


Fig.3.10.-Rutinas del menú Edit

Actualmente, aunque en el algoritmo principal se refleja el código de estas funciones, no están activadas, como podemos observar en la figura,debido a errores de programación en el código fuente.Es una de las mejoras futuras que se plantean en el algoritmo, con el fin de sacarle el máximo rendimiento a la aplicación.

Menú Partition:

Este es uno de los menús más elaborados,de hecho es la finalidad del proyecto, ya que todo circula entorno a la partición y a sus posibles ediciones y modificaciones.

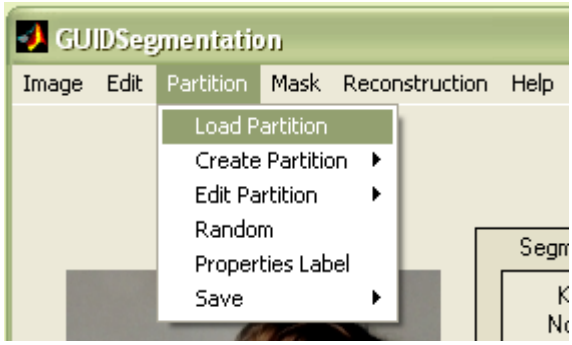


Fig.3.11.-Rutinas *menú Partition*

a).-Item Load Partition

Como se ha comentado en la sección de *Descripción* de la aplicación, existe la posibilidad de trabajar directamente con particiones ya generadas guardadas en disco, desde el *ítem Load Partition*.Un tipo de particiones con las que podemos trabajar es con las particiones generadas por un algoritmo de segmentación proveniente de Soft-Image, software de procesado de imagen del grupo de imagen del Departamento de Teoría de la Señal y Comunicaciones.

Es un algoritmo implementado en el lenguaje de programación C, suya función principal, que se adjuntará al apéndice de este documento, es llamada ***segintra_model0(args, 1)***. Se hará un breve comentario sobre el funcionamiento de dicha función.

funcionamiento de programación :

Segintra_modelo0(args,1).-

El primer argumento es asociado a la estructura de C que almacena todos los datos necesarios para realizar la segmentación. El segundo de hecho es asociado al empleo del miembro "vop_in" de la estructura 'args'. Si el miembro "vop_in" es cero no tiene efectos sobre el resultado.Por otro lado, si "vop_in" tiene un valor concreto, la segmentación es realizada sólo dentro del grupo de regiones especificadas por la imagen "vop_in". El argumento de barrido se usa para llamar a la función de etiquetado del grupo de regiones que no son segmentadas en la partición de salida. Si el argumento se pone a 0 (cero),la cambia el valor de todas aquellas regiones que estaban anterirormente por la etiqueta 0 (El cero). Sin embargo,si el argumento es diferente del cero, el algoritmo de etiquetado es usado etiquetar las últimas regiones que han entrado en la cola por la correspondiente etiqueta para crear la imagen final.

El resultado es una imagen en la cual, cada componente conexa está etiquetada con un valor diferente.

VER para mayor información otras funciones complementarias,almacenadas en el SOFT-IMAGE: segintra_model0_conf_file(), segintra_model0_gray(), segintra_model0_color().

Para el correcto funcionamiento de esta función, ha sido necesaria la elaboración de otras funciones que complementan a esta, para llevar a cabo la segmentación.Funciones que no se van a describir, pero a las que se hace alusión para despertar la curiosidad del lector en el estudio de las mismas.

En la figura 3.12.- se muestra el resultado de este algoritmo de segmentación, para una segmentación de la imagen de 100 regiones.

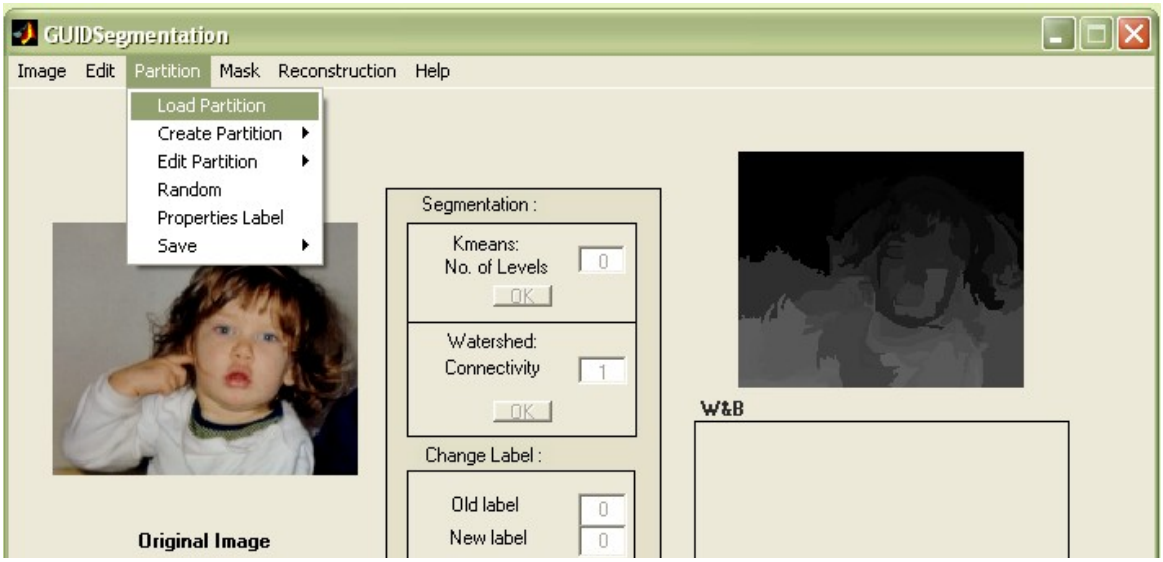


Fig.3.12.-Partición de la imagen de entrada, cargada desde disco, generada por el algoritmo de Luís Garrido para el software SOFT-IMAGE.

b).-Item Create Partition:

Para esta aplicación podemos hablar de dos técnicas de segmentación,el Algoritmo por clustering ***k-means*** y el Algoritmo por métodos de inundación ***Watershed.***

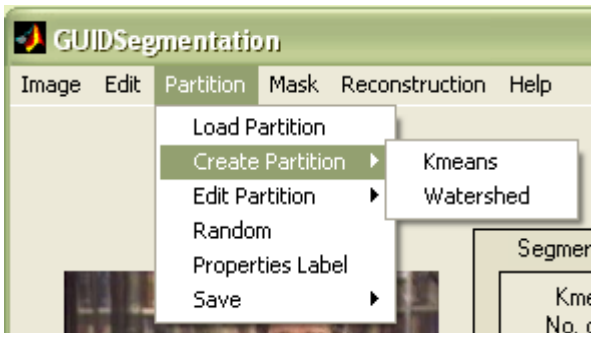


Fig.3.13.-Rutinas ítem Create Partition

K-means: La idea básica del Algoritmo es obtener los K centros iniciales y formar clusters asociando todos los objetos de X a los centros más cercanos,después recalcular los centros.
La función utilizada para la llamada de este algoritmo en nuestro software es la siguiente:

```
function [mu,out]=kmeans(ima,k)
```

Las variables asociadas son:

Input:
ima: imagen en niveles de grises
k: centros iniciales,

Output:
mu: vector de clusters,

Para la entrada de estos parámetros se ha utilizado las llamadas cajas editables del Guide de Matlab,la entrada de esta String será almacenada en el handle editable y convertiva a valor numérico para su posterior utilización como parámetro de entrada en la llamada a la función kmeans.La imagen de entrada, es la imagen original convertida en niveles de grises.

Una vez seleccionado el ítem, se activará de forma automática en la aplicación, la sección de Segmentación,pero sólo aquellos parámetros relacionados con el Kmeans.Introducimos el valor de los clusters en la caja editable y presionamos el botón *'OK'*.Como se muestra a continuación en la figura.

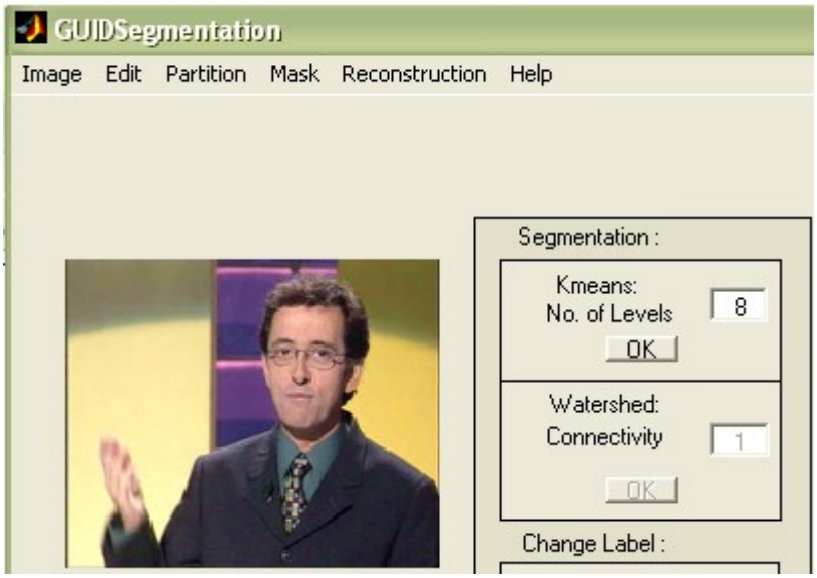


Fig.3.14.-Parametros para el Algoritmo Kmeans

La función para el Algoritmo de Segmentación, *Kmeans*, es programada como función externa,es independiente de la guide principal, se puede apreciar bajo estas líneas el código fuente de la misma.

function [mu,out]=kmeans(ima,k)

Trabajamos con la imagen en niveles de gris,

```
ima=rgb2gray(ima);
```

se chequea la imagen,

```
copy=ima;          % make a copy
ima=ima(:);      % vectorize ima
mi=min(ima);       % deal with negative
ima=ima-mi+1;      % and zero values
s=length(ima);
```

posterirormente se crea el histograma de la imagen,

```
m=max(ima)+1;
h=zeros(1,m);
hc=zeros(1,m);

int im;
im=0;

for i=1:s
        im= ima(i);
          if  im >0,
                h(ima(i))=h(ima(i))+1;
          end
 end
```

```
ind=find(h);
hl=length(ind);
```

se inizializan los centroides,

```
mu=(1:k)*m/(k+1);
```

En esta parte del algoritmo inicializamos el proceso de clustering

```
while(true)
oldmu=mu;
```

clasificación actual de centroides,

```
        for i=1:hl
                c=abs(ind(i)-mu);
                cc=find(c==min(c));
                 hc(ind(i))=cc(1);
         end
```

y finalmente se hace una recalculación de los clusters a utilizar.

```
         for i=1:k,
                  a=find(hc==i);
                mu(i)=sum(a.*h(a))/sum(h(a));
         end

         if(mu==oldmu)
               break;
         end;
  end
```

La máscara obtenida será la partición resultante

```
s=size(copy);
mask=zeros(s);
for i=1:s(1),
        for j=1:s(2),
                c=abs(copy(i,j)-mu);
                a=find(c==min(c));
                mask(i,j)=a(1);
        end
end

mu=mu+mi-1;   %recovered real

out=mask;
```

El proceso para la llamada a la función Watershed es similar, pero en está ocasión, como ya se ha comentado anteriormente la función empleada pertenece a la Toolbox de Procesado de Imagen de Matlab. La conectividad empleada para las pruebas de la aplicación ha sido 26-conn puesto que la entrada en una imagen de color verdadero RGB.

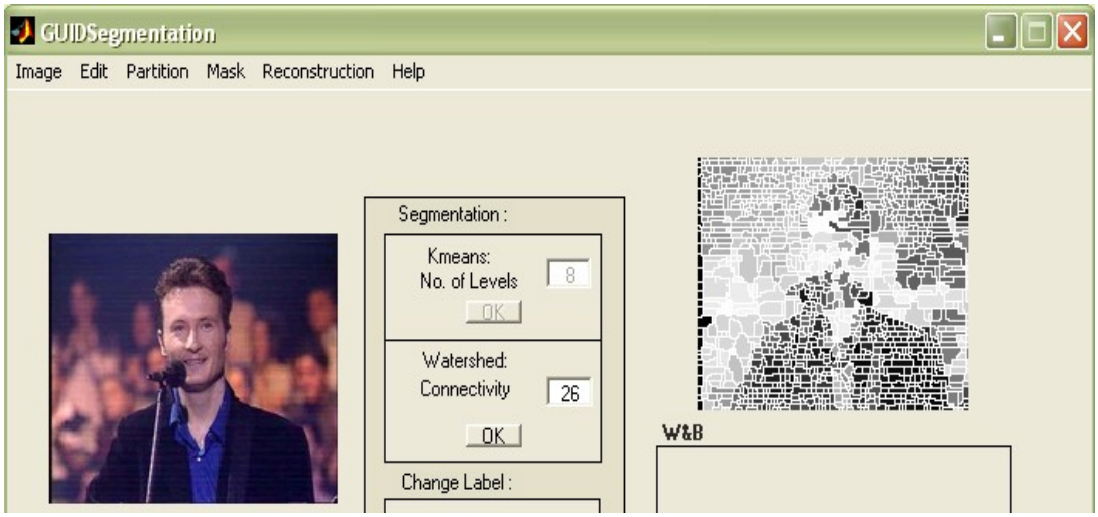


Fig.3.15.-Partición por el método de inundación Watershed

Siguiendo el recorrido dentro del menú de Partición, nos encontramos varias funciones sobre edición de la partición, como es conocer el valor de una o varias etiquetas de la región de interés para el usuario.

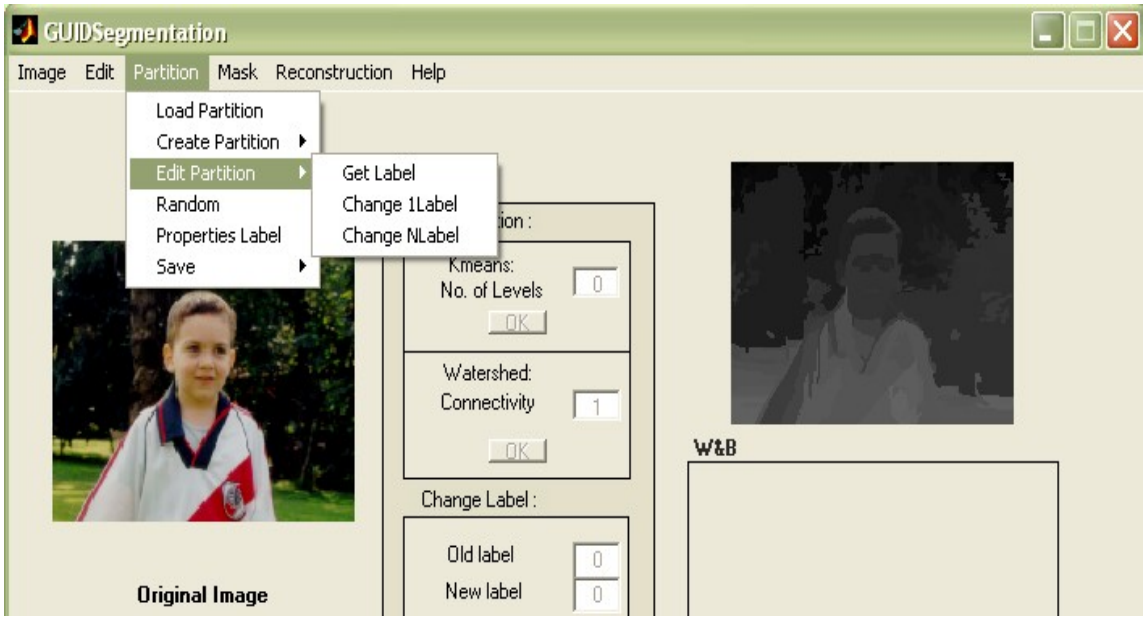


Fig.3.16.-Rutinas item Edit Partition

c).-Item GetLabel:

La función Get Label, está programada dentro del guide principal,bajo el nombre de: *function* ***LabelPartition_Callback****(hObject, eventdata, handles),* nos devuelve una ventana de diálogo denominada *Color picker*, en el que se muestran todos los valores de las etiquetas clicadas con el ratón.También podemos acceder a ella desde el botón de la aplicación '*GetLabe*l'.

Para conseguir estos valores de entrada desde el ratón, se ha utilizado una función del Toolbox de Procesado de Matlab como herramienta.

[c, r, P] = impixel;

Impixel: Determina el valor de los pixeles seleccionados con el ratón o con el teclado, y los almacena en el vector P.

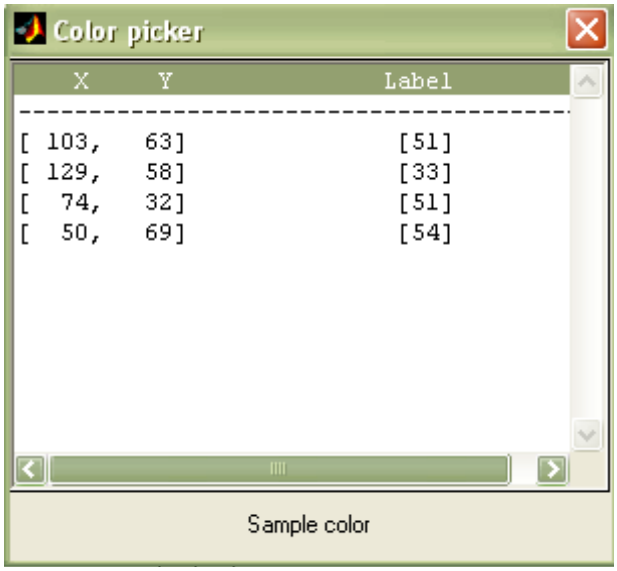


Fig.3.17.-ventana diálogo etiquetas seleccionadas en la partición

d).-Item Change Label /Change Nlabel:

Las funciones ***change1Label*** *y* ***changeNlabel****,* nos dan la opción de cambiar el valor de una de las etiquetas de la partición, o varias,introducien como parámetros el valor de la etiqueta/s a cambiar y el nuevo valor por el que queremos sustituirla/s.Estas, también son funciones callback, que se producen al presionar el botón *'Ok* ' de la sección *Change Labels* de la aplicación. Estas funciones callbacks llaman a otras funciones externas nombradas *changelabel y changeNlabel,* respectivamente.

Se utiliza la misma estructura guide para ambas llamadas, la diferencia es que en el caso de ***changeNlabel***, las etiquetas son varias y se tomarán desde la ventana de diálogo del *color picker*, como hemos comentado anteriormente. Puesto que los valores de entrada se toman desde esta ventana emergente, la caja editable correspondiente a 'old label' aparecerá desactivada.

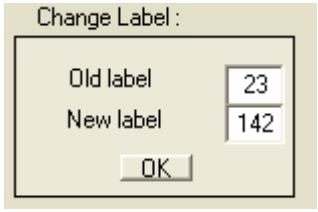


Fig.3.18.-Entrada de parámetros para cambio de etiquetas.

Los callbacks a las funciones son:

```
function SelectLabel_Callback(hObject, eventdata, handles);
function SelectNLabel_Callback(hObject, eventdata, handles);
```

Los handles utilizados para estas funciones son: handles.oldlabel, handles.newlabel, handles.accept.

Si el *handles.oldlabel* está activado indica que la llamada es a la función *change1label* (cambiar una única etiqueta) de lo contrario los valores entran desde la ventana de diálogo y el handles.oldlabel está activado, por lo que el algoritmo llamará a la función changeNlabel.

```
if (handles.oldlabel)
        old=get(handles.oldlabel,'Value');
        new=get(handles.newlabel,'Value');
        set(handles.part,part);

        ima_out=changelabel(part,old,new);
else
  Pto=handles.P; Donde P es un vector que guarda todos los valores de las etiquetas
  seleccionadas  por el ratón.
  new=get(handles.new,'Value');
  ima_out=changeNlabel(part,Pto,new);
end
```

changelabel: recorre la matriz de la imagen partición y cambia aquellos píxeles que tienen valor '*old*' por el valor '*new*'.

```
function out=changelabel(ima,old,new)
  imacol=ima;
  [f,c]=size(imacol);

  for i=1:f
     for j=1:c

        if imacol(i,j)==old;
           imacol(i,j)=new;
        end
    end
  end
   out=imacol;
```

changeNlabel: la función es similar a la anterior, ahora en vez de tener un valor de la etiqueta a cambiar, tenemos un vector *'P'* con varias etiquetas a modificar.

```
function out=changeNlabel(image,P,new)

    imacol=image;
    [f,c]=size(imacol);
    %P is a vector
    N=length(P);


    for k=1:N
        v=P(k);
            for i=1:f
                    for j=1:c

                    if imacol(i,j)==v,
                       imacol(i,j)=new;


                    else
                       imacol(i,j);
                    end
                end
        end
    end

     out=imacol;
```

e).-Item Random:

El botón *Random* nos ayudará a visualizar esta partión ,en niveles de gris, en color. El *ítem Random* está asociado a la función *aleatorio*, al igual que el botón. La llamada a esta función hace uso de la herramienta matemática de Matlab, como la función rand() ó cat().

rand(n,m): -genera matrices nxm de entradas aleatorias entre 0 y 1.
cat(DIM,A,B):-concatena las matrices A y B dando como resultado una matriz de dimensión DIM.

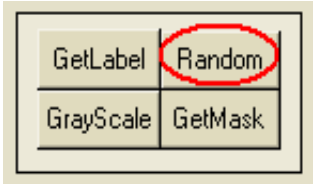


Fig.3.19.-Botón Random

Código Fuente:

```
function out=aleatorio(part)
        part=double(part);
        N=max(max(part));

        %Colour Lut table
        LutR=zeros([(N+1),1]); %vector of N rows
        LutG=zeros([(N+1),1]);
        LutB=zeros([(N+1),1]);

        [m,n]=size(part);

        for i=1:N,

                LutR(i)=rand(1);
                LutG(i)=rand(1);
                LutB(i)=rand(1);
        end

        part_r=double(zeros(size(part)));
        part_g=double(zeros(size(part)));
        part_b=double(zeros(size(part)));


        for i=1:m,
          for j=1:n,

                value=part(i,j);



                if value==0

                        part_r(i,j)= 0;
                        part_g(i,j)=0;
                        part_b(i,j)=0;
                else
                          r=LutR(value,1);  part_r(i,j)= r;
                          g=LutG(value,1); part_g(i,j)=g;
                          b=LutB(value,1);  part_b(i,j)=b;
                 end
             end
         end

         out=cat(3,part_r,part_g,part_b);
```

Si presionamos varias veces sobre el botón de Random obtendremos múltiples particiones de color aleatorio sin que existan repeticiones de las mismas. En la *Fig.3.20* se muestran algunos ejemplos,

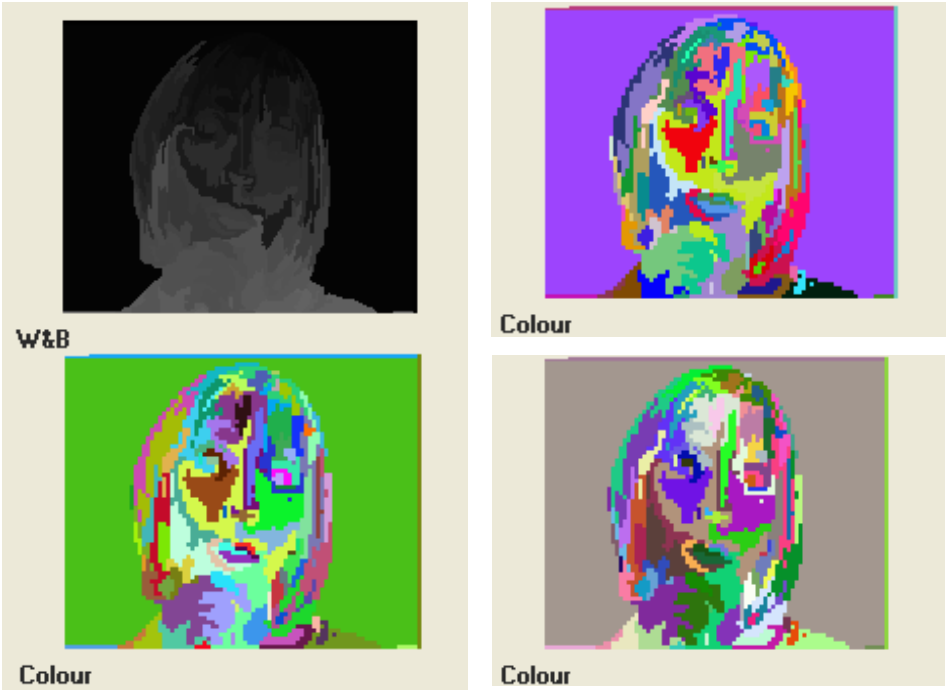


Fig.3.20.-Particiones color aleatorias

f).-Item Properties of the Labels:

Haciendo uso nuevamente de herramientas de Matlab, la aplicación da la opción de conocer las propiedades de las etiquetas seleccionadas,mediante la función regionprops explicada en el apartado de Descripción general, en este mismo capítulo del documento.

Fig.3.21.- Devuelve las propiedades de la etiqueta seleccionada

La función guarda la información en la variable stats, que es un vector con longitud igual al número de etiquetas de la partición y para cada región contiene todas las propiedades de la misma.Por ejemplo,para una partición cuya etiqueta seleccionada corresponde a la región 23, de las 100 regiones que contiene la partición, obtendríamos la siguiente salida:

Ej.-La región etiquetada como 23, contiene 50 píxeles.

```
          Area: 50
      Centroid: [118.9200 55.1600]
   BoundingBox: [114.5000 49.5000 12 10]
   Subarrayldx: {1x2 cell}
MajorAxisLength: 13.8124
MinorAxisLength: 7.0322
  Eccentricity: 0.8607
   Orientation: -34.3132
    ConvexHull: [16x2 double]
   ConvexImage: [10x12 logical]
    ConvexArea: 82
         Image: [10x12 logical]
   FilledImage: [10x12 logical]
    FilledArea: 50
   EulerNumber: 1
       Extrema: [8x2 double]
 EquivDiameter: 7.9788
      Solidity: 0.6098
        Extent: 0.4167
 PixelIdxList: [50x1 double]
    PixelList: [50x2 double]
```

Fig.3.22.-Propiedades de las regiones de la partición

Menú Mask:

a).-Item Create Mask:

Para la creación de la máscara,dentro del ítem Mask, nos hemos basado en la función de changeNlabel ya que el procedimiento es similar, la diferencia entre ambas funciones es,que las etiquetas entrantes serán cambiadas por valor '255', y el resto de las etiquetas se pondrán a '0', de manera que la máscara será una imagen binaria de 0's y 1's.

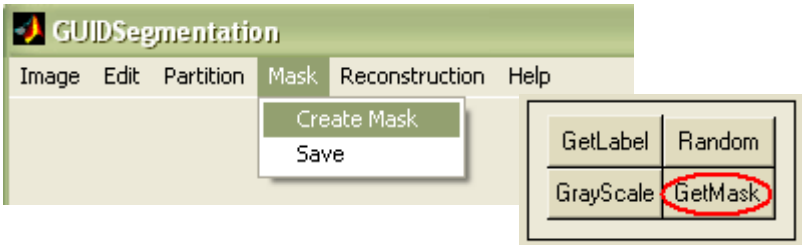


Fig.3.23.-Menú Mask>item Create Mask
>Botón GetMask

Al presionar con el ratón sobre el item *Create Mask* ó el botón *GetMask*, el icono del ratón se convertirá en una cruz permitiendo al usuario clicar sobre aquellas regiones de interés para la creación de la máscara, una vez seleccionadas pulsa la tecla de 'enter', se abrirá una ventana emergente para que a modo de reconfiramción,el usuario presione el botón *Ok* ,para empezar con el proceso de crear máscara.

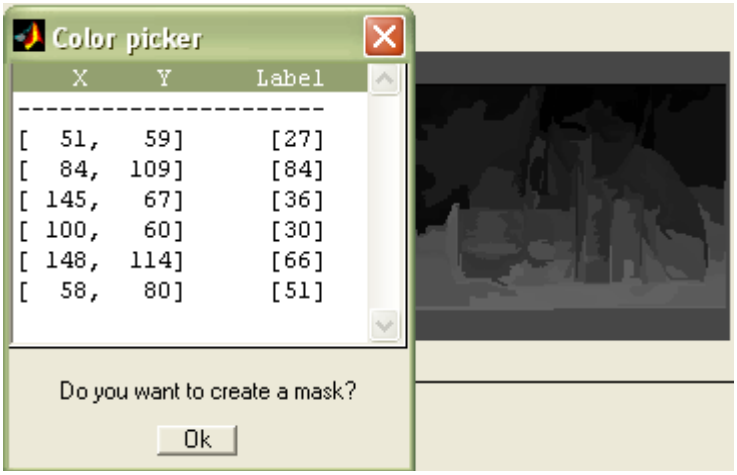


Fig.3.24.- Ventana emergente confirmación de máscara

Domask: la función recorre la partición buscando aquellas etiquetas que se corresponden con los valores almacenados en el vector 'P',poniendo los píxeles de la etiqueta a 255 y el resto a 0.La salida de la función es la máscara.

```
function img_mask=Domask(part, P)

    mask=zeros(size(part));
    [f,c]=size(part);
    %P is a vector
    N=length(P);


    for k=1:N
        v=P(k);
            for i=1:f
                for j=1:c

                    if part(i,j)==v,
                       mask(i,j)=255;
                    else
                        mask(i,j);
                    end

                end
            end
      end
   img_mask=mask;
```

b).-Item Save:

Tenemos la posibilidad al igual que en el caso de la partición de salvar la máscara visualizada en el eje *'mask'* en la parte inferior derecha de la aplicación.

Menú Reconstruction:

Esta imagen binaria de ceros y unos será utilizada en la reconstrucción de la imagen principal.Esta reconstrucción es represetada a través de esta máscara o de su invertida.

a).-Item Rec. Normal
b).-Item Rec. Inverted

Para acceder a la reconstrucción, en el *menú Reconstruction*, nos situaremos en el *item Rec. Normal ó Rec. Inverted*. En la Reconstrución los valores de la máscara que están a 255 serán sustituidos en la imagen de salida por los valores de la imagen original, y los valores de la máscara que son 0 se mantendrán a 0. En la Reconstrucción invertida,se procederá de manera inversa.

Los valores que en la máscara valen 255 en la salida serán 0, y los valores 0 en la imagen de salida tomarán el valor de la imagen original en ese píxel.

Fig.3.25.-Item Reconstruction

Código fuente de la función de Reconstrucción:

```
function img_rec=Rec(mask, img)

    img_rec=zeros(size(img));
    img=double(img);                %convert to double because the mask is double array
    [f,c]=size(img);
    [r,col]=size(mask);


    for i=1:f
      for j=1:c

          if isgray(img),
                    for k=1:r
                for l=1:col
                     if mask(k,l)==255,
                       img_rec(i,j)=img(i,j);
                          else
                       img_rec(i,j)=0;
                        end
            end
```

```
                end

            else
                if isrgb(img),
                        for k=1:r
                           for l=1:col
                                if mask(k,l)==255,
                                   Rimg_rec(i,j)=img(i,j);
                                   Gimg_rec(i,j)=img(i,j);
                                   Bimg_rec(i,j)=img(i,j);
                                else
                                  Rimg_rec(i,j)=0;
                                   Gimg_rec(i,j)=0;
                                   Bimg_rec(i,j)=0;
                           end
                             img_rec=cat(3,Rimg_rec,Gimg_rec,Bimg_rec);
                      end
                   end
                end
             end

          end
       end


    out=img_rec;
```

c).-Item Save:

La reconstrucción de la imagen original será visualizada en uno de los ejes y podrá salvarse en disco desde el menú Save del menú de Reconstrucción.

Por ejemplo, como resultado final visualizaremos los siguiente:

Fig.3.26.-Resultado de la Reconstrucción

Menú Help:

El último menú a comentar es el Help,es una nueva guide con información sobre el contenido del proyecto en rasgos generales. Una vez dentro de la guide siempre puedes volver al menú principal a través del botón GUIDSegmentation.

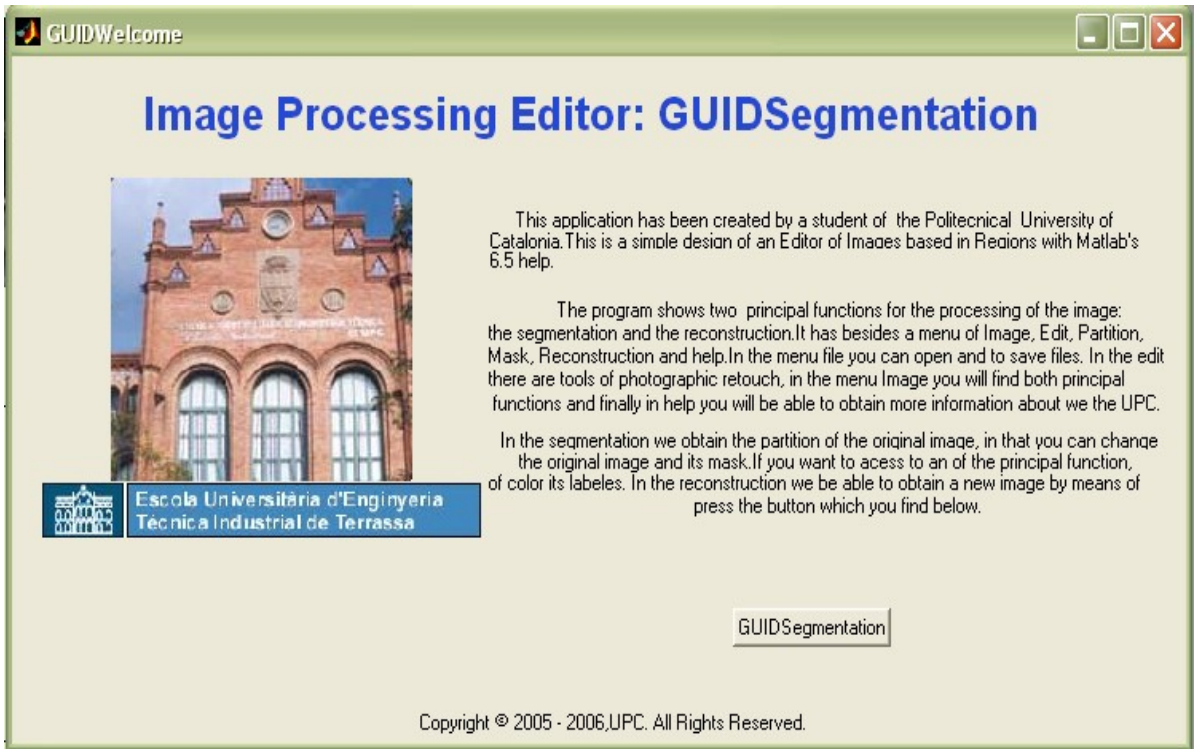


Fig.3.27.-Guide Help

CAPÍTULO IV
CONCLUSIONES,CONTRIBUCIONES Y LÍNEAS DE TRABAJO FUTURAS

Conclusiones, Contribuciones y líneas de Trabajo futuras.-

4.1.-Introducción.-

Debido a la naturaleza aplicada del proyecto que se presenta, se han enfocado los esfuerzos hacia el desarrollo de una aplicación en funcionamiento. Esto supone un especial cuidado de los aspectos de programación de la aplicación y la necesidad de indagar en diversos campos dentro de la teoría de la segmentación, así como en otros de programación en Java, al inicio del proceso y de entorno matlab, y en menor medida,a la computación en tiempo real; porque no se ha prestado demasiada atención a este punto.

De hecho, una considerable parte de las referencias encontradas en el texto corresponden a resultados provenientes de tesis doctorales o anotaciones del Departemento de Teoría de la Señal y Comunicaciones, bien de la Upc o de otras universidades dedicadas a este campo, que cubren aspectos parciales de alguna de las diversas problemáticas involucradas.

No obstante, surgen ocasiones en las que las soluciones propuestas en la literatura para ciertos problemas parciales resultan inapropiadas o incompletas, al menos para la aplicación en cuestión. En estos casos ha sido necesario profundizar en los mismos en busca de una solución o de un mejor entendimiento de la limitaciones y restricciones existentes.

Este capítulo se centra en sintetizar las conclusiones y contribuciones que el autor cree más relevantes y en indicar posibles líneas de investigación relacionadas con los objetivos globales del proyecto.

4.2.-Conclusiones.-

En esta sección se sintetizan las conclusiones o resultados que el autor ha considerado más relevantes dentro del ámbito de aplicación del sistema desarrollado y de los objetivos perseguidos con el mismo.

La elección de un espacio de representación adecuado para el color sigue constituyendo un reto para el procesado y análisis de las imágenes color.La segmentación es un concepto clave en el tratamiento digital de la imagen.La segmentación de una imagen implica la detección, mediante procedimientos de etiquetado deterministas o estocásticos, de los contornos o regiones de la imagen, basándose en la información de intensidad o/ y la información espacial.

Para imágenes a nivel de gris,una de las técnicas más clásicas y simples de segmentación supone la determinación de los modos o agrupamientos (''clusters'') del histograma,lo cuál permite una clasificación o umbralización de los píxeles en regiones homogéneas. Este método, como hemos visto, se puede generalizar para imágenes multiespectrales, teniendo en cuenta que la segmentación de las imágenes en color implica inicialmente la elección de un espacio de representación para el color.

Hemos presentado en este documento métodos morfológicos para la segmentación de imágenes en niveles de gris, como el K-means y el Watershed, otros para imágenes en color como el algoritmo de Luís Garrido y Philippe Salembier. El procedimiento tomado se ha basado en la conjunción de una umbralización automática de la variable de saturación (dualidad acromáticao vs. Cromático) y de un criterio conectivo de clasificación morfológica, mediante agrupamiento de puntos en un espacio 2D, para segmentar el histograma, como en el caso del *K-means.* A su vez, también se ha mostrado la segmentación mediante métodos morfológicos que se fundan en la línea divisora de aguas, el Watershed.

➔ Bajo las condiciones de trabajo experimentadas a las que ha sido sometido la interficie desarrollado en este proyecto, cabe pensar que la aplicación de técnicas basadas en el algoritmo de Luís Garrido, para la estimación de la partición, es la más factible.

➔ Para ello, se hace necesario realizar un código de programación que enlace el entorno matlab con el nivel bajo de programación C, en versiones futuras. Con el fin de poder trabajar directamente con las particiones obtenidas en el proceso de la segmentación sin tener que hacer uso de la base de datos de imágenes almacenadas en disco.

➔ La representación que ofrece este algoritmo proporciona una sólida base para la edicción de la partición en nivel de gris y para la manipulación de la información estimada en la misma. Su representación es más apropiada, que la obtenida mediante el algoritmo de clasificación K-mean o Watershed.

➔ Para la utilización de la partición obtenida mediante el proceso de división por líneas de agua sería necesaria trabajar en profundidad en el algoritmo con el fin de eliminar los contornos blancos, que limitan las distintas regiones, para facilitar la edicción de la partición.

Resultados de Watershed

➔ Sería positivo también para la interficie conseguir un correcto funcionamiento de las herramientas de edicción propuestas, como el *zoom* o el *crop*, en el que nos permetirían segmentar una zona determinada de la imagen sin necesidad de segmentarla en su totalidad.

➔ Con el hardware adecuado y con el empleo de algortimos enfocados al aprovechamiento de éste, es posible cumplir los requisitos de tiempo real deseados, ya que muchas de las funciones utilizadas con llevan un tiempo de computación elevado entorpeciendo el funcionamiento correcto de la aplicación. La utilización de lenguajes de programación en un nivel de programación más bajo como C, también sería una buena solución a este problema.

4.3.-Contribuciones.-

La intención de esta sección es la de resaltar aquellas contribuciones que el autor considera más relevantes en cuanto a su influencia en la viabilidad del sistema, a su novedad dentro de la disciplina en la cuál está englobada o a su concordancia con los objetivos iniciales.

➔ Estudio del tema de segmentación de imágenes como parte fundamental en el proceso de procesado de imagen.

➔ Desarrollo de una aplicación en entorno Matlab, que permite crear y editar particiones, como resultado de la aplicación de un algoritmos de segmentación a la imagen original.

➔ Se ha desarrollado un algoritmo original que permite:

1. el coloreo aleatorio de las regiones de la partición.Facilitándonos una mejor visualización de las regiones de la misma.

2. modificar los valores de las etiquetas de la partición.

3. crear la máscara a partir de la combinación de regiones de la partición.

4. y reconstruir la imagen original dentro de los límites de la máscara.

Reconstrucción de la imagen original

4.4.-Líneas de trabajo futuras.-

La percepción tridimensional de entornos empleando técnicas de procesado de imagen aún está en los albores de su consagración como procedimiento de aplicación industrial o para la medica. Por ello, el número de posibilidades y caminos es inmenso. Adicionalmente, el carácter aplicado de este proyecto conllevado a pensar en diversos campos, cada uno de los cuales ofrecen un amplio abanico de aspectos por investigar y mejorar.

En esta sección se indican algunas acciones de investigación futura que el autor estima interesantes y posibles abordar a corto o medio plazo:

- ➔ Incorporar el algoritmo de segmentación basado en crecimiento de regiones de Luís Garrido y Philippe Salembier, comentado en numerosas ocasiones.

- ➔ Incorporar otros algoritmos de segmentación, que no se han mencionado en este documento, pero que pueden ser de gran utilidad.

- ➔ Mejorar la edicción de etiquetas de manera que te permita además de fusionar dos regiones distintas de una partición, la separación de una región en dos subregiones mediante una línea divisoria o cuenca.

APÉNDICE

Apéndice.-

El algoritmo de Segmentación de Luís Garrido ha sido de gran uso para esta aplicación, ya que todas las particiones recogidas en la carpeta de imágenes adjunta al proyecto, son particiones resultantes de su software.

Ya que ha sido un algoritmo de tanto interés para la interface, se ha decido en versiones futuras ampliar la parte de segmentación añadiendo este algoritmo, programado desde C. La conexión entre ambos entornos de programación, se realiza a través de las llamadas funciones MEX. Debido a la falta de tiempo, no se han podido incorporar estas funciones al algoritmo principal, pero se aporta como documentación extra el código Mex, pero en formato.c sin compilar como archivo Mex, necesario para la llama a estas funciones en C desde Matlab.

```
        /*****************************************************************************/
        /************************** ***INCLUDE FILES ********************************/

#include <string.h>

#include "segment_mex.h"
#include "image.h"
#include "matlab_util.h"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
  int i;
  int iscolor;
  int nreg;
  double psnr;
  struct image ima_in,ima_s;

  /* Process all Options */
  if (nrhs<1) {
        mexEvalString("help segment");
        mexErrMsgTxt("At least two input arguments required");
  }

  i=0;
    if (nrhs>=(i+1)) {
        /* first argument is the number of region segmented */

        if (mxIsDouble(prhs[i])==false) {
          mexErrMsgTxt("Second argument must be a scalar.");
        }

        iscolor = (long) mxGetScalar(prhs[i]);
  }

  if (nrhs>=(i+1)) {
        /* second argument is the input  image */

        /* check for an image */
        if (mxIsUint8(prhs[i])==false) {
          mexErrMsgTxt("First argument must be an uint8 image.");
        }
```

```
        /* check for a double image */

        if (mxIsDouble(prhs[i])==false) {
          mexErrMsgTxt("First argument must be a double image.");
        }

        /* Copy images to struct images */
        mxArray2image( prhs[i], &ima_in);

 }
  i++;

if (nrhs>=(i+1)) {
        /* third argument is the number of region segmented */

        if (mxIsDouble(prhs[i])==false) {
          mexErrMsgTxt("Second argument must be a scalar.");
        }

        nreg = (long) mxGetScalar(prhs[i]);

 }

i++;

 if (nrhs>=(i+1)) {
        /* third argument is the number of psnr*/

        if (mxIsDouble(prhs[i])==false) {
          mexErrMsgTxt("Second argument must be a scalar.");
        }
        psnr = (long) mxGetScalar(prhs[i]);

 }

 if (nrhs>=(i+1)) {
        /* second argument is the input  image */

        /* check for an image */
        if (mxIsUint8(prhs[i])==false) {
          mexErrMsgTxt("First argument must be an uint8 image.");
        }

        /* check for a double image */
        if (mxIsDouble(prhs[i])==false) {
          mexErrMsgTxt("First argument must be a double image.");
        }

        /* Copy images to struct images */
        mxArray2image( prhs[i], &ima_s);

 /* Create space for the output image */
 timmalloc( &ima_in, &ima_s);


 /* Call the skeleton function in Soft_Image */
segment_image( &iscolor, &ima_in,&nreg,&psnr,&ima_s);
```

```
/* Copy output image to matlab */
  image2mxArray( &ima_s, &plhs[0]);

  /* Free structs */
  imfree( &ima_in);
   imfree( &ima_s);

}
```

La librería .h, llamada desde la función segment_mex.c se codifica de la siguiente manera:

```
function out=segment_Image(iscolor,ima, nreg,psnr);

% SEGMENT_IMAGE calculates the partition of an image for the original image
%
%     img = segment_Image(iscolor,ima, nreg,psnr)

%     "ima" is the input image.
%      "nreg" is the number of the regions that the user wants to segment
%      original image
%     "psnr" is the floating point
%     "iscolor" is true if the image is rgb
%
%
%            Samira Hervella Azouzi (26/05/2006)


out=segment_mex(iscolor,ima, nreg,psnr);
```

Bibliografía.-

Referencias Literarias.-

[Apuntes] Asignatura de Procesado de Imagen, EUETIT.

[ArtículoI] Region-based representations of image and video:Segmentation tools for multimedia services.P. Salembier,F.Marquez. IEEE Transactions on circuits and systems for video Technology, vol. 9, Nº 8. Dec 1999 (pages 1147-1167).

[ArtículoII] Image Sequence analysis and Merging algorithms. Philippe Salembier, Luis Garrido, David García. Universitat Politécnica de Catalunya.Campus Nord, Módulo D5. C/Gran Capitá, sn. 08950 Barcelona.Spain.

[ArtículoIII] Segmentación de imágenes en color utilizando histogramas en espacios color. *Revista "Computación y Sistemas", Vol. 8, no. 4, June 2005.*

[González] Digital Image Processing Using MATLAB (Hardcover) by Rafael C.González, Richard E.Woods, Steven L. Eddins.

[Handbook] The Matlab handbook Eva Pärt-Enader, Anders Sjoberg, Bo Melin, Pernilla Isaksson.Harlow, England [etc.] Addison- Wesley 1996.

[León] Procesamiento Digital de Imagen fundamentos y prácticas con Matlab de Enrique Alegre Gutiérrez... Universidad de León,Secretariado de Publicaciones.

Referencias Web.-

[aisa] www.aisa.uvigo.es

[teleco] www.teleco.uvigo.es

Comentario.-

La utilización de estas páginas web es debida, a mi primer año de carrera realizado en la universidad de Vigo, ETS Enxeñeiros de Telecomunicacions.Me resulta de mucha utilidad la información adquirida a través de esta web, así como la consulta de apuntes y tesis que facilitan los estudiantes.